uc3m | Universidad Carlos III de Madrid

University Degree in Computer Science & Engineering
Academic Year 2017-2018

Bachelor Thesis

"UNIVERSAL ACCESS TO UML MODELS USING KNOWLEDGE MANAGEMENT. OSLC KM CONNECTOR"

Alejandro Al Maullem Jarones

Juan Bautista Llorens Morillo

José María Álvarez Rodriguez

TABLE OF CONTENTS

1. INTRODUCTION 8

1.2 OBJECTIVES 9

2. STATE OF THE ART 10

2.1 SEMANTIC WEB STACK 10

3. ANALYSIS 12

3.1 USER STORIES 12

3.2 USE CASES 13

3.3 USER REQUIREMENTS 15

Table 7: User Requirement 6 16

3.4 SYSTEM REQUIREMENTS 17

3.4.1 Functional Requirements 17

3.4.2 Non-Functional Requirements 22

3.5 REQUIREMENT DEPENDENCY MATRIX 25

3.6 REQUIREMENT TRACEABILITY MATRIX 26

4. DESIGN 27

4.1 TECHNOLOGY 27

4.2 METHODOLOGY 27

4.3 SYSTEM CONTEXT DIAGRAM 28

4.4 CLASS DIAGRAM 29

4.5 PACKAGE DIAGRAM 30

4.6 SEQUENCE DIAGRAMS 31

Figure 15: **Error! Bookmark not defined.**

4.7 COMPONENT DIAGRAM 35

4.8 COMPONENT DESCRIPTION 35

4.9 COMPONENT TRACEABILITY 40

4.10 DEPLOYMENT DIAGRAM 41

4.11 DATA TRANSFORMATION 42
4.12 USER INTERFACE 46
4.12.1 LOAD MENU 46
4.12.2 EXPORT MENU 46
5. IMPLEMENTATION & TESTING 48
5.1 SUPPORT TOOLS USED 48
5.2 METHODOLOGY 49
5.3 SPRINTS 49
5.3.1 SPRINT ONE 50
5.3.2 SPRINT TWO 50
5.3.3 SPRINT THREE 50
5.3.4 SPRINT FOUR 50
5.3.5 SPRINT FIVE 51
5.3.6 SPRINT SIX 51
5.4 TESTING 52
5.4.1 TEST CASES 52
5.4.2 TEST TRACEABILITY MATRIX 58
6. BUDGET AND PLANNING 59
6.1 TIME PLANNING 59
6.2 GANTT DIAGRAM 62
6.3 BUDGET BREAKDOWN 63
6.3.1 Direct Costs 63
6.3.2 Indirect Costs 64
6.3.3 Budget Summary 65
7. SOCIO-ECONOMIC ENVIRONMENT AND LEGAL FRAMEWORK 66
7.1 ECONOMIC IMPACT 66
7.2 LEGAL IMPLICATIONS 66
7.2.1 MAGICDRAW LICENSING 66
7.2.2 ECLIPSE LICENSE 67

7.2.3 NOTEPAD++ LICENSE 67

8. CONCLUSIONS AND FUTURE WORK 68

8.1 FUTURE WORK 69

9. BIBLIOGRAPHY 70

INDEX OF FIGURES

Figure 1: OSLC data model

Figure 2: Semantic Web Stack 2005 and Semantic Web Stack 2015

Figure 3: MagicDraw user use case

Figure 4: Undefined user use case

Figure 5: System context diagram

Figure 6: Class diagram

Figure 7: Package diagram

Figure 8: Load diagram sequence diagram

Figure 9: Request all diagrams sequence diagram

Figure 10: Request project diagrams sequence diagram

Figure 11: Request current diagram sequence diagram

Figure 12: Request diagram by ID sequence diagram

Figure 13:Request diagram by name sequence diagram

Figure 14: Export diagram to file sequence diagram

Figure 15: Component diagram

Figure 16: Deployment diagram

Figure 17: Load menu mockup

Figure 18: Export menu mockup

Figure 19: Notepad++ JSTool tree view

INDEX OF TABLES

Table 1: User Stories

Table 2: User Requirement 1

Table 3: User Requirement 2

Table 4: User Requirement 3

Table 5: User Requirement 4

Table 6 User Requirement 5

Table 7: User Requirement 6

Table 8: System Functional Requirement 1

Table 9: System Functional Requirement 2

Table 10: System Functional Requirement 3

Table 11: System Functional Requirement 4

Table 12: System Functional Requirement 5

Table 13: System Functional Requirement 6

Table 14: System Functional Requirement 7

Table 15: System Functional Requirement 8

Table 16: System Functional Requirement 9

Table 17: System Functional Requirement 10

Table 18: System Functional Requirement 11

Table 19: System Functional Requirement 12

Table 20: System Functional Requirement 13

Table 21: System Functional Requirement 14

Table 22: System Functional Requirement 15

Table 23: System Functional Requirement 16

Table 24: System Non-Functional Requirement 1

Table 25: System Non-Functional Requirement 2

Table 26: System Non-Functional Requirement 3

Table 27: Requirement Dependency Matrix

Table 28: Requirement Traceability Matrix

Table 29: Description of Component 1

Table 30: Description of Component 2

Table 31: Description of Component 3

Table 32: Description of Component 4

Table 33: Description of Component 5

Table 34: Description of Component 6

Table 35: Component Traceability Matrix

Table 36: SRL-MagicDraw Elements Transformation

Table 37: Test Case 1

Table 38: Test Case 2

Table 39: Test Case 3

Table 40: Test Case 4

Table 41: Test Case 5

Table 42: Test Case 6

Table 43: Test Case 7

Table 44: Test Case 8

Table 45: Test Case 9

Table 46: Test Case 10

Table 47: Test Case 11

Table 48: Test Traceability Matrix

Table 49: Project Hours Computation

Table 50: Gantt Diagram

Table 51: Human Resources Costs

Table 52: Consumables Costs

Table 53: Direct Costs Summary

Table 54: Budget Summary

GLOSSARY

AI: Artificial Intelligence

API: Application Programming Interface

FR: Functional Requirement

GSON: Google Search Organization Network

IoT: Internet of Things

JSON: JavaScript Object Notation

MD: MagicDraw

NFR:Non Functional Requirement

OASIS: Organization for the Advancement of Structured Information Standards

OSLC: Open Services for Lifecycle Collaboration

OSLCKM: Open Services for Lifecycle Collaboration Knowledge Management

OWL: Ontology Web Language

RDF: Resource Description Framework

RDFS: Resource Description Framework Schema

REST: Representational State Transfer

RIF: Rule Interchange Format

RSHP: RelationSHip, pronounced arship. It is a knowledge representation model.

SR: System Requirement

SRL: System Representation Language

UR: User requirement

1. INTRODUCTION

Nowadays, in all processes involved in engineering, many types of artifacts are created. Some examples of such artifacts are requirements, models, test cases, between others. Furthermore, there many tools that are used in these contexts, each with their own format for data representation, as well as different communication protocols.

This situation generates a preeminent problem: the transit of data between different systems. This can be a challenge to affront at times, since a good structure to provide a solution to this problems offers engineers the possibility of reusing many hours of work already done.

Some systems such as OSLC already provide some solutions to confront this issue by providing a way to exchange data between systems, nevertheless, this solutions will not provide a standardized data representation. Furthermore, these tools are not able to properly export all the information contained in the different programs used, and will not provide any form of way of exporting the operations that they produce. These operations are important to enable their reuse. Also, some of these operations allow the validation of the data contained in the artifacts, providing a much needed data integrity.

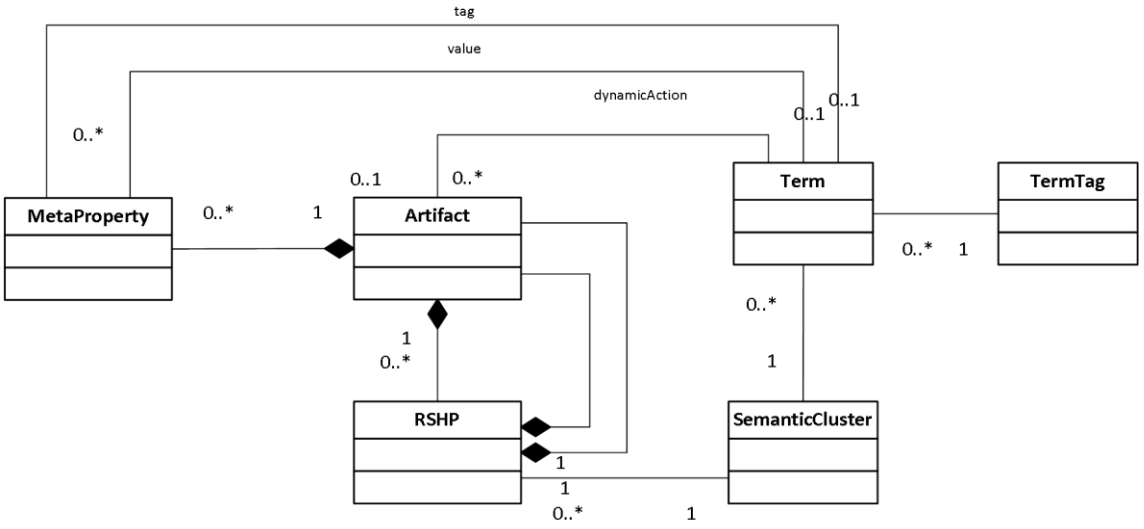

Figure 1: OSLC data model [1]

A proper solution to solve the presented issue will need to provide a way to reuse all data included in the artifact in hand in each case (or at least as much as possible) as well as the operation the tool being used includes, through interoperability. These operations must be available through some provided interfaces to make it possible to reuse them.

The objective of this project is to provide a tool that will be able to interpret and transform data and operations created by MagicDraw, one of the mentioned engineering tools available, into OSLC format in a way that will provide a way to use it in different environment.

1.2 OBJECTIVES

Through the realization of this project there are several objectives that are aspired to be reached, in order to improve my knowledge in the area of the developed application.

During this project it is an objective to study the current state of knowledge management in the scope of engineering. It is a new area to be explored for me, related in great measure with the studies that I have been developing in the past years.

It is also a purpose of this project to analyze the current state of interoperability between engineering-focused tools, making distinction between their different flavour. During this process their weaknesses and strengths are aimed to be found, to provide a view of the necessities of the current engineering industries that are not being met with the current technologies.

After obtaining the current status and its flaws, it is intended to, basing myself on the existing technologies, design a solution that will contribute to the development of a solution to the existing situation. This solution will need to provide a way of transforming engineering artifacts into a common data format that will enable the reuse of data and operations throughout different systems. In this way the solution will provide not only a way to transfer information into other systems, but also access loads of information from other sources.

More concretely, it is intent of this venture to create a working implementation to transform the data included in any MagicDraw diagram into the OSLC representation data. Furthermore, it will need to be able to provide an interface of a OSLCKM consumer to obtain this diagram, as well as be able to use existing OSLCKM interfaces to obtain external data, interpreting it and converting it appropriately to a MagicDraw diagram to be viewed and edited by the user at will.

After a successful implementation of the proposed system, it will be imperative to make sure the developed implementation follows quality standards and has the expected functionality. This is necessary to be able to offer this solution as a way to transfer data, since it would be useless if there is no guarantee that the data sent/received has any kind of accuracy.

2. STATE OF THE ART

Many different approaches have been taken into standardization of data to seek the ability to provide interoperability between different tools. More concretely, two main approaches of semantic networks have been proposed to represent different kinds of data.

2.1 SEMANTIC WEB STACK

The semantic web stack is a hierarchy of different data representations, used to represent different types of knowledge. These representations are organized in the way shown in figure 2.

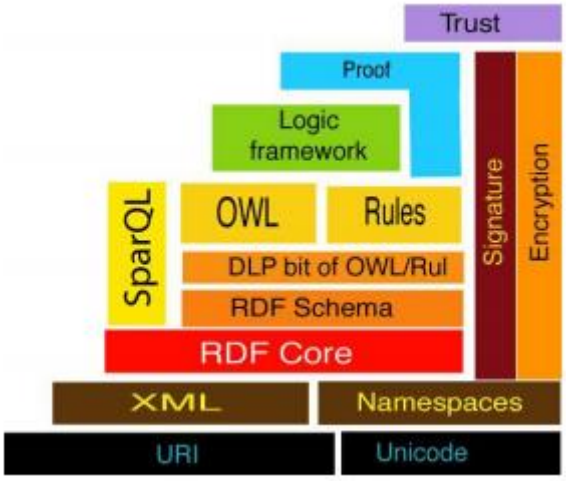

Figure 2: Semantic Web Stack [2]

The different languages that compose this structure are the following:

- RDF[3]: It is a common and shared data model that represents web resources in the shape of a directed graph. It is an abstract representation form, containing three main elements:
    - Subject
    - Object
    - Predicate

  Each triplet produced with these elements is denominated an RDF graph. These graphs can be visualized as a set of nodes connected by directed arcs, establishing a directed graph.

- RDFS[4]: RDF Schema is a modeling vocabulary for RDF data. Id defines classes and its properties and a domain and range for each of these properties.
- OWL[5]: It is a vocabulary used to define formal ontologies. Comprised inside this language, we can find different sublanguages:
    - Expressions Language
    - Query Language
    - Rule Language

- RIF[6]: This format is a set of dialects used to exchange business rules. The different dialects it compromises are:
    - Logic based dialects
    - Rules with Actions
- RSHP[7]: This is ra representation system that allows all kinds of information to be represented using a conceptual graph model. It is based on the assumption that any type of data can be described as a set of concepts related between each other.

While all these methods have their own advantages, they also present some detriments.

RDF is composed of two elements: resources and literals, but presenting the disadvantage that literals cannot be set as the subject of the RDF triplet. Not only that, but there is a lack of tools that directly manipulates RDF. It also presents some advantages; it is a great way of representing ontologies, and can be serialized as RDFS. Caution must be taken with this last one, since RDFS is only recommended for use with web resources related statements.

OWL provides a logic framework that can be used for any kind of data. It also provides a way to check consistency, but there are no validation methods. Besides, interpretation of data cannot be done without a big performance loss.

RIF was never design to include data to be validated, and also has very few tools that can process it. Nevertheless, RID was created with the objective to share data through the we, which gives it an advantage against others.

RSHP will allow the creation of relationships, that will associate artifacts, concepts and terms, providing a way of representing data with any cardinality through its relationships. This enables a high level of expressivity when using this framework. There is also a bigger amount of support tools available. The main disadvantage of this method is the fact that it is not meant to be used in a web environment, and it has to be adapted for it.

OSLC[8] is an initiative created to standardize data in the interest of creating interoperability between different tools used in the software engineering environment. It is an increasing effort o find a new way to integrate these tools easily, and allowing the creation of a better development and operations process. It is now an OASIS standard, for tools that are involved in software engineering.

OSLC has made good advancements in the interoperability sector, still has some restrictions regarding data representation, specially with relationships with different cardinalities.

All of these alternatives permit to some extent the standardization of data from different environments(being OSLC the most appropriate one), nevertheless none of them provide a true interoperability, since they are not able to collect all implicated data and they do not expose any interface to reuse the operations that can be carried out to said data.

3. ANALYSIS

3.1 USER STORIES

First of all, the user stories will be defined to show the functionalities that will be desired by the user. These user stories will have the following format:

$$As\ a\ <WHO>\ I\ want\ <WHAT>\ so\ that\ <WHY>$$

These user stories will be on a high level, in a way a user could describe their necessities, and will be expanded upon in the following sections of this document.

As a...	I want...	So that...
MagicDraw User	My MagicDraw diagrams to be available to OSLCKM.	Obtain the OSLC representation of a diagram in SRL.
User	Retrieve MagicDraw diagrams' information through OSLCKM	Have access to the information the diagrams convey
MagicDraw User	To import a diagram in SRL from an OSLCKM URL.	Access diagram data from different origins.

Table 1: User Stories

In the table above two different users have been defined: a MagicDraw user and a non specified user. The MagicDraw user will, as its name implies, be the user that uses Magicdraw; and the unspecified user will be one that wants to access MagicDraw information from outside the program. This users may end up being the same physical person in practice.

3.2 USE CASES

The use cases will describe the different activities the user will execute along the use of the different functionalities the system provides.

On figure 4, the use case for the MagicDraw user is presented. In this use case the user is able to create a new diagram or edit an existing one through the native MagicDraw functionality. Doing this the diagram is automatically available for requests from an OSLCKM consumer to retrieve and reuse the information of the diagram, this is the reason this use case is included even though it is a standard MD functionality. On the other hand this user may also load a diagram from an OSLCKM service provider, by opening the load menu, inserting the URL and OAUTH settings. Then he will be able to interact with the new diagram. The MagicDraw user may also export the current diagram to a JSON file.

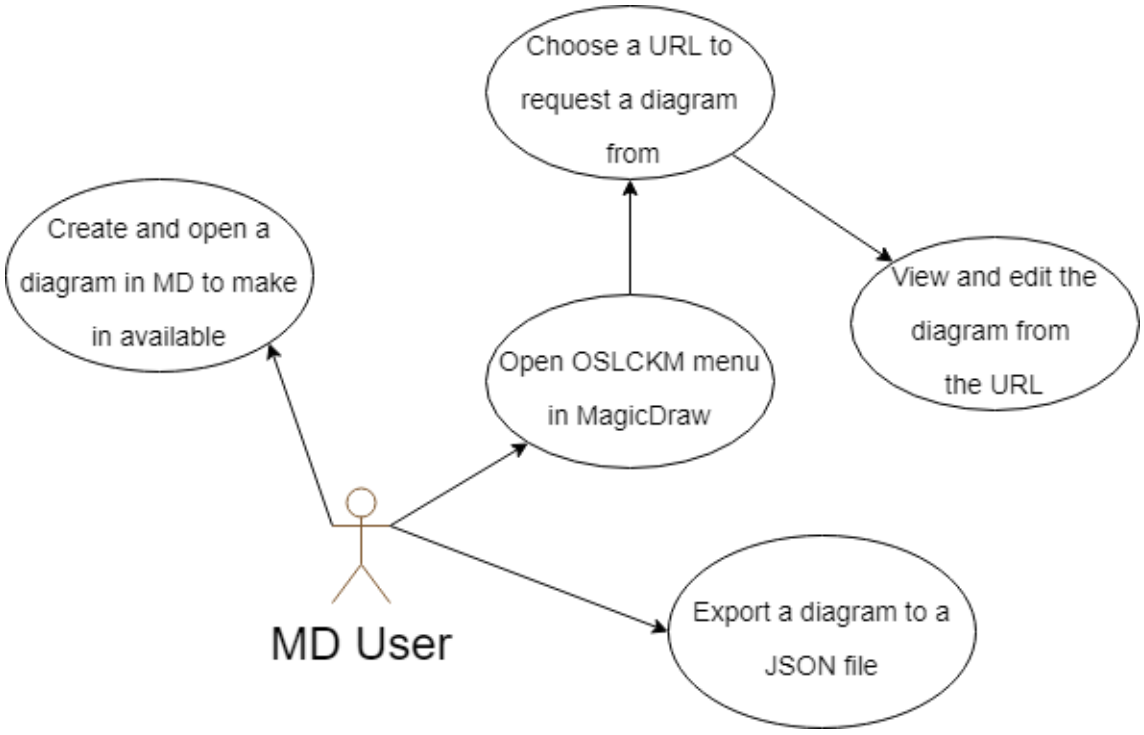

Figure 3: MagicDraw user use case

In figure 5 we can see the second use case, this time for an external user. This user will request from MagicDraw a diagram through the OSLCKM interface provided. He can choose to request a specific diagram through its name or id, or choose to retrieve them all(at project or global level). In order for this user to be able to access certain diagram the MD user must have the project the diagram belongs to open, since there is no way to access the diagram if not.

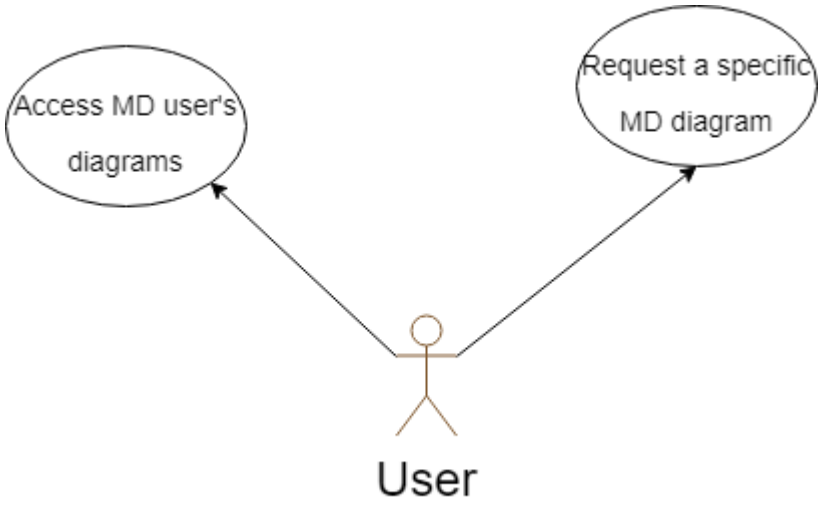

Figure 4: Undefined user use case

3.3 USER REQUIREMENTS

From the user stories and user cases defined above, the user requirements can be determined. The stories will be divided into different desirable outcomes the user expects in each of the functionality paths each story follows.

ID	UR-01		
Priority	High	Necessity	Essential
Clarity	High	Verifiability	High
Description	The user shall be able to connect his/her MagicDraw diagrams to the OSLC paradigm, allowing its information to be processed correctly.		

Table 2: User Requirement 1

ID	UR-02		
Priority	High	Necessity	Essential
Clarity	High	Verifiability	High
Description	The user shall be able to view two new buttons to open a menus with the new functionality options.		

Table 3: User Requirement 2

ID	UR-03		
Priority	High	Necessity	Essential
Clarity	High	Verifiability	High
Description	The user shall be able to choose the URL from which the diagram is imported from.		

Table 4: User Requirement 3

ID	UR-04		
Priority	High	Necessity	Desirable
Clarity	High	Verifiability	High
Description	The user shall be able to know the imported diagram is consistent.		

Table 5: User Requirement 4

ID	UR-05		
Priority	High	Necessity	Essential
Clarity	High	Verifiability	High
Description	The user shall be able to view and edit the diagram from the requested URL.		

Table 6: User Requirement 5

ID	UR-06		
Priority	High	Necessity	Desirable
Clarity	High	Verifiability	High
Description	The user shall be able to export the current diagram into a JSON file in SRL format.		

Table 7: User Requirement 6

3.4 SYSTEM REQUIREMENTS

3.4.1 Functional Requirements

Below, the functional requirements the implemented system must comply with are defined. To ease the understanding of them, they will be presented in a tabular manner. Each of the field in the table will be defined:

- ID: Identifier used for the requirement, plus a descriptive name for it. The id will follow the pattern SR-FR_XX, being X the number of the requirement.
- Priority: How urgent it is for the functionality to be implemented.
- Necessity: How much value this functionality provides to the final user of the system.
- Clarity: How understandable the requirement is.
- Verifiability: How easy it is to verify the requirement has been fulfilled.

ID	SR-FR_01 Receive request for current diagram		
Priority	High	Necessity	Desirable
Clarity	High	Verifiability	High
Description	The system shall be able to receive a request through a URL to obtain the SRL for the current diagram.		

Table 8: System Functional Requirement 1

ID	SR-FR_02 Receive request for current project		
Priority	Medium	Necessity	Desirable
Clarity	High	Verifiability	High
Description	The system shall be able to receive a request through a URL to obtain the SRL for all the diagrams in the current project.		

Table 9: System Functional Requirement 2

ID	SR-FR_03 Receive request for all diagrams		
Priority	High	Necessity	Essential
Clarity	High	Verifiability	High
Description	The system shall be able to receive a request through a URL to obtain the SRL for all open diagrams.		

Table 10: System Functional Requirement 3

ID	SR-FR_04 Receive request for specific diagram(ID)		
Priority	Medium	Necessity	Desirable
Clarity	High	Verifiability	High
Description	The system shall be able to receive a request through a URL to obtain the SRL for a diagram identified through its ID.		

Table 11: System Functional Requirement 4

ID	SR-FR_05 Receive request for specific diagram(Name)		
Priority	Medium	Necessity	Desirable
Clarity	High	Verifiability	High
Description	The system shall be able to receive a request through a URL to obtain the SRL for a diagram identified through its Name.		

Table 12: System Functional Requirement 5

ID	SR-FR_06 Parse diagrams		
Priority	High	Necessity	Essential
Clarity	Medium	Verifiability	Medium
Description	The system shall transform MagicDraw diagrams to SRL, to later display through the web services stated in SR-FR_01 through SR-FR_05.		

Table 13: System Functional Requirement 6

ID	SR-FR_07 Filter diagrams by ID		
Priority	Medium	Necessity	Desirable
Clarity	High	Verifiability	High
Description	The system shall filter through all diagrams by ID.		

Table 14: System Functional Requirement 7

ID	SR-FR_08 Filter diagrams by Name		
Priority	Medium	Necessity	Desirable
Clarity	High	Verifiability	High
Description	The system shall filter through all diagrams by Name.		

Table 15: System Functional Requirement 8

ID	SR-FR_09 Display two buttons in the MagicDraw toolbar		
Priority	High	Necessity	Essential
Clarity	High	Verifiability	High
Description	The system shall display two new buttons in the MagicDraw toolbar labeled “Load SRL Diagram” and “OSLCKM Export Diagram”.		

Table 16: System Functional Requirement 9

ID	SR-FR_10 Show the load menu		
Priority	High	Necessity	Essential
Clarity	High	Verifiability	High
Description	The system shall display a pop-up menu upon clicking the first button specified in SR-FR_09. This menu must allow the user to choose a URL from where to request the diagram in SRL format.		

Table 17: System Functional Requirement 10

ID	SR-FR_11 Check URL format validity		
Priority	High	Necessity	Essential
Clarity	High	Verifiability	High
Description	The system shall validate the URL inserted by the user in the menu defined in SR-FR_10 is a proper URL (has URL format).		

Table 18: System Functional Requirement 11

ID	SR-FR_12 Request SRL diagram		
Priority	High	Necessity	Essential
Clarity	High	Verifiability	High
Description	The system shall request a diagram in SRL format from the URL typed by the user as specified in SR-FR_10.		

Table 19: System Functional Requirement 12

ID	SR-FR_13 Parse SRL diagram		
Priority	High	Necessity	Essential
Clarity	Medium	Verifiability	Medium
Description	The system shall transform the SRL diagram obtained in SR-FR_11 into a MagicDraw Diagram, saving it to the current project.		

Table 20: System Functional Requirement 13

ID	SR-FR_14 Check diagram validity		
Priority	High	Necessity	Essential
Clarity	Medium	Verifiability	Medium
Description	The system shall validate the diagram created in MagicDraw to ensure it is consistent.		

Table 21: System Functional Requirement 14

ID	SR-FR_15 Show export menu		
Priority	High	Necessity	Desirable
Clarity	Medium	Verifiability	High
Description	The system shall show the export menu when the second button specified in SR-FR_09 is clicked.		

Table 22: System Functional Requirement 15

ID	SR-FR_16 Export diagram to file		
Priority	High	Necessity	Desirable
Clarity	Medium	Verifiability	High
Description	The system shall save the current diagram in SRL to a file.		

Table 23: System Functional Requirement 16

3.4.2 Non-Functional Requirements

Below, the non functional requirements the implemented system must comply with are defined. To ease the understanding of them, they will be presented in a tabular manner. Each of the field in the table will be defined:

- ID: Identifier used for the requirement, plus a descriptive name for it. The id will follow the pattern SR-NFR_XX, being X the number of the requirement.
- Priority: How urgent it is for the functionality to be implemented.
- Necessity: How much value this functionality provides to the final user of the system.
- Clarity: How understandable the requirement is.
- Verifiability: How easy it is to verify the requirement has been fulfilled.

ID	SR-NFR_01 Response Time		
Priority	High	Necessity	Essential
Clarity	High	Verifiability	Medium
Description	The maximum response time of the system shall be 100ms per processed diagram.		

Table 24: System Non-Functional Requirement 1

ID	SR-NFR_02 MagicDraw version		
Priority	High	Necessity	Essential
Clarity	High	Verifiability	High
Description	The plugin shall be compatible for MagicDraw 19.0.		

Table 25: System Non-Functional Requirement 2

ID	SR-NFR_03 Interface and usability		
Priority	High	Necessity	Essential
Clarity	Medium	Verifiability	Medium
Description	The interface described in SR-FR_10 shall be user friendly, easing usability.		

Table 26: System Non-Functional Requirement 3

3.5 REQUIREMENT DEPENDENCY MATRIX

The requirement dependency matrix will compare and display if any requirement is dependent on another one.

In the matrix we can observe how the requirements of the menus depend on the ones of showing the buttons, since the buttons are needed to open the menu. The load menu requirement is necessary itself in order to fulfil the validation of the URL, since it is not possible without the interface to recollet said URL. The same scenario occurs with the export menu and the exporting to the file inserted, the export cannot be realized without the file path collected from the interface.

We can also see a dependency between the requests received with specific diagrams, and the requirements that filter the diagrams, since they cannot be filtered until a filtering parameter is received.

	SR-FR_1	SR-FR_2	SR-FR_3	SR-FR_4	SR-FR_5	SR-FR_6	SR-FR_7	SR-FR_8	SR-FR_9	SR-FR_10	SR-FR_11	SR-FR_12	SR-FR_13	SR-FR_14	SR-FR_15	SR-FR_16
SR-FR_1																
SR-FR_2																
SR-FR_3																
SR-FR_4							X									
SR-FR_5								X								
SR-FR_6																
SR-FR_7				X												
SR-FR_8					X											
SR-FR_9										X	X	X			X	
SR-FR_10									X		X					
SR-FR_11									X	X		X				
SR-FR_12									X		X		X			
SR-FR_13												X		X		
SR-FR_14													X			
SR-FR_15									X							X
SR-FR_16															X	

Table 27: Requirement Dependency Matrix

3.6 REQUIREMENT TRACEABILITY MATRIX

The requirement dependency matrix will match the system requirements defined with the user requirements, to ensure all user necessities are met.

In the matrix we can see how the first user requirement is covered by all the first system requirements, which define the different web export operations. The second one deals with the user interfaces used. User requirements three, four and five are covered by the requirements in charge of loading a valid diagram, the main functionality of the program.Finally, the functionality of exporting to a file corresponds 1:1 from user to system requirements.

	UR-01	UR-02	UR-03	UR-04	UR-05	UR-06
SR-FR_1	X					
SR-FR_2	X					
SR-FR_3	X					
SR-FR_4	X					
SR-FR_5	X					
SR-FR_6	X					
SR-FR_7	X					
SR-FR_8	X					
SR-FR_9		X				
SR-FR_10		X	X			
SR-FR_11			X			
SR-FR_12			X		X	
SR-FR_13				X	X	
SR-FR_14				X	X	
SR-FR_15		X				
SR-FR_16						X

Table 28: Requirement Traceability Matrix

4. DESIGN

The architectural design of the developed application will be explained in this section. First, the technology decisions taken will be explained, followed by the design methodology chosen to disclose the elected architectural design. Then the architectural design will be exposed. Finally, the method of transforming information from MagicDraw to SRL format will be described.

4.1 TECHNOLOGY

MagicDraw provides an open API of free use in order to implement plugins that can interact with its models and add new functionalities. Since MagicDraw is a program running in the Java Virtual Machine, these plugins are also implemented in Java.

Therefore, the most appropriate technology to use in order to implement the required functionalities will be a plugin implementing Magicdraw's openAPI, which will be written in Java.

This API will also allow the addition of new elements to the interface, such as toolbar menus (as required in this case). Therefore using the openAPI it will be possible through the provided implementations of Plugin and MDAction to create all the required functionality.

4.2 METHODOLOGY

The system's design will be done using Unified Modeling Language (UML)[9]. A method that, using a graphical approach, following the saying "An image is worth more than a thousand words". With this method the design and documentation of the system can be represented.

Since this application has most of its functionality in the background, there is not much interface use (Only one simple menu is used). Therefore, there is not much sense in considering a Model-View-Controller architecture.

The model that has been chosen, taking into account the special characteristics of this system will be the Master-Slave pattern. This way, the master will be the MagicDraw Plugin instance that will call all other functionalities(slaves) when needed.

4.3 SYSTEM CONTEXT DIAGRAM

In the diagram below, the different interactions of the plugin to be implemented with external systems is explained. Through the interface provided by MagicDraw's openAPI, the plugin will be able to display a new menu and access and edit the models opened in the program. Through the OSLCKM interface, the diagrams may be sent and received, making available the diagrams provided by OSLCKM consumers to the MagicDraw user.

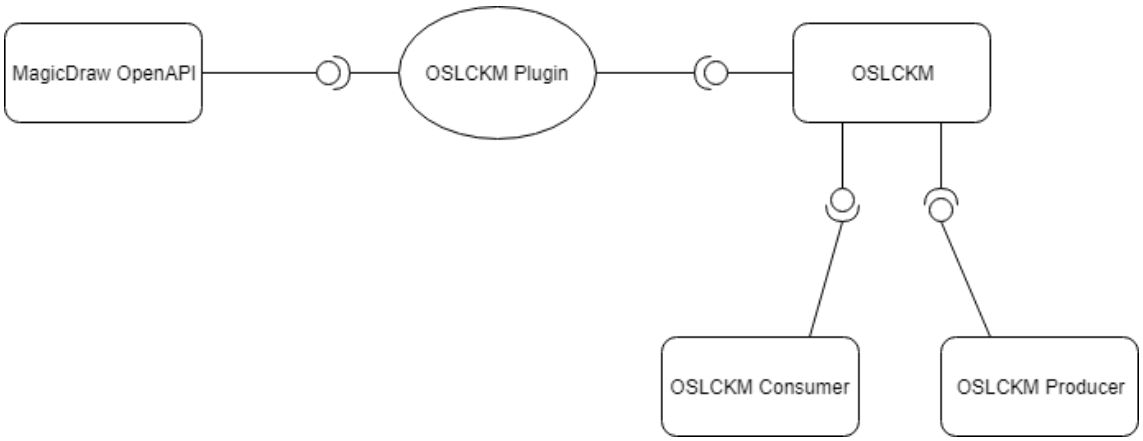

Figure 5: System context diagram

4.4 CLASS DIAGRAM

Below, the class diagram is represented to aid in the understanding of the architectural classes organization.

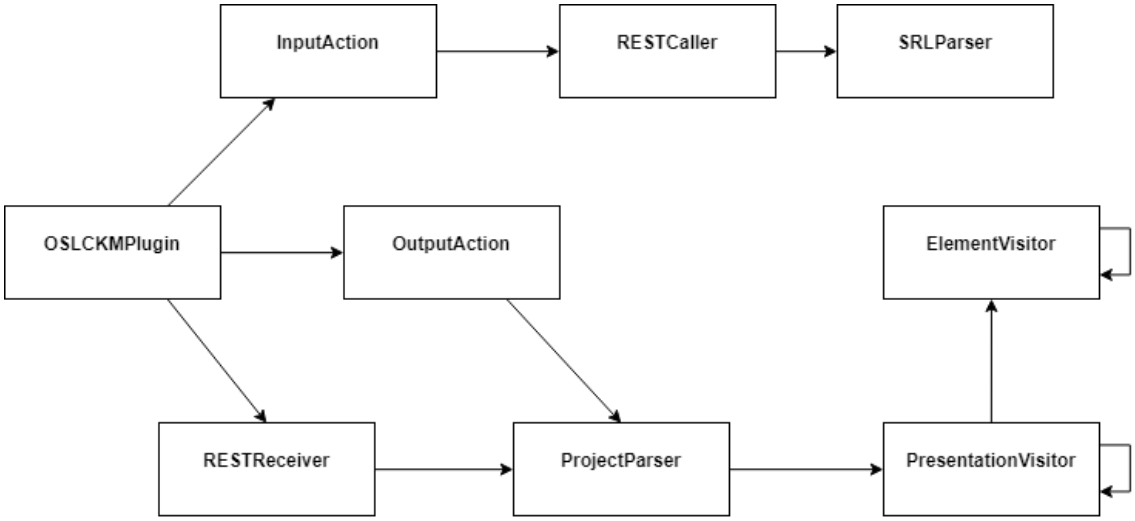

Figure 6: Class diagram

A brief definition of the classes and their main functionalities (in a high level) will be provided to ensure full understanding of the structure and responsibilities of each of the classes:

- OSLCKMPlugin: This class will extend MagicDraw's openapi Plugin class. It will initialize the component starting the REST service that will listen to incoming requests and calling InputAction to create the toolbar menu.
- RESTReceiver: This class will be responsible of listening at a url for requests sent inquiring for diagrams.
- ProjectParser: This class is responsible for retrieving diagrams from MagicDraw, as well as filtering them when necessary.
- Visitors[10]: Here, both clases used the design pattern of a visitor. This class organization model is meant to represent on elements of the same structure, in this case the diagram being processed. This technique allows to operate on existing classes, in a way that the code is organized and well divided. In this case two different visitors are used to visit different types of elements, since it is the classes provided by MagicDraw: one will be used for presentation elements and the other for element instances.
    - PresentationVisitor: the PresentationVisitor will extend MagicDraw's openapi Visitor class. It will be responsible of parsing presentation elements for one diagram at a time and transforming them into an Artifact that contains all the diagram's information.
    - ElementVisitor: This class will also extend a visitor, ModelHierarchyVisitor in this case. It will parse proper elements instead of presentation elements.

- InputAction: Extends MDAction. It will be responsible of creating the loading menu the user must use and ensure the inserted URL is valid.
- InputAction: Extends MDAction. It will be responsible of creating the file creation menu for the user.
- RESTCaller: Will create REST requests against the URL inserted by the user to obtain a SRL diagram in JSON format.
- SRLParser: This class will transform the SRL elements into a diagram and insert it into MagicDraw.

4.5 PACKAGE DIAGRAM

Below is depicted the structure of the packages of code that will be created. As stated previously, a Model-View-Controller organization is not being followed in this system, because of the characteristics of it. Therefore, the packages will need a different organization as the one that would be used in these cases.

Since all operations can be divided in two generic services, the packages will divide the classes in two packages, one for inputting and one for outputting data. One third package will be created with common classes that are used to initialize the plugin and interact with OSLCKM.

This structure also represents the Master-Slave pattern implemented, being the input and output both the slaves of the common package.

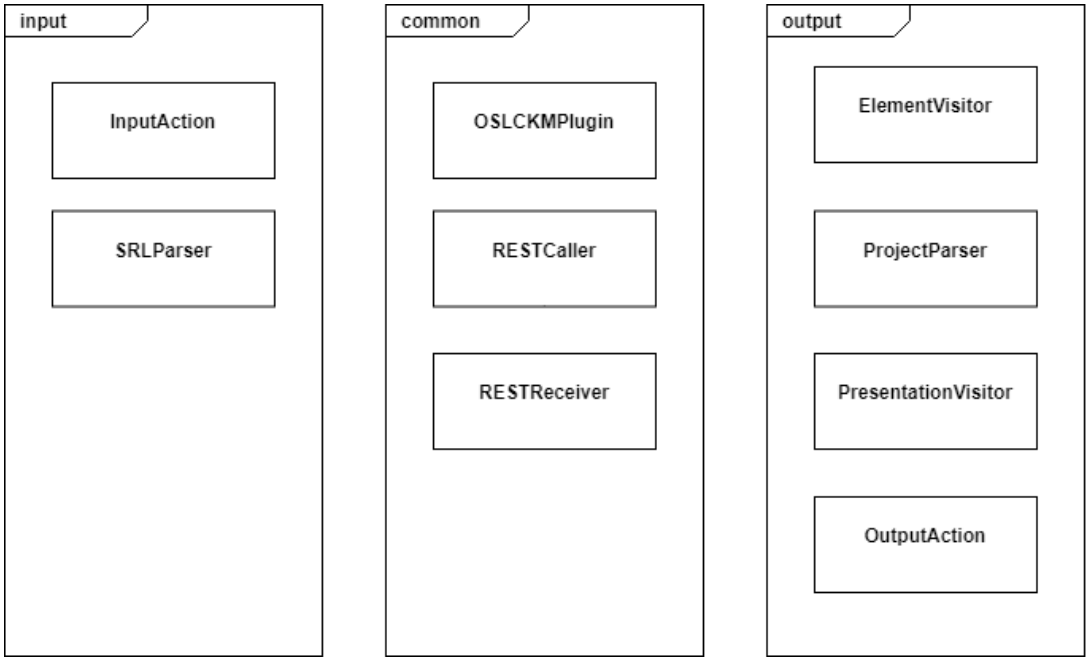

Figure 7: Package diagram

4.6 SEQUENCE DIAGRAMS

Below, the sequence diagrams with the different functionality routes the program will go through with different user interactions will be depicted. In all these diagrams, the action is initialized by the user using MagicDraw or by the unspecified user stated in the use cases. For the later case, “OSLCKM” has been used to represent this interaction.

On figure 8 the different steps through the classes to load a diagram form the OSLCKM interface is shown. This is one of the most complete operations, considering the use of the classes involved is specific only to this functional path.

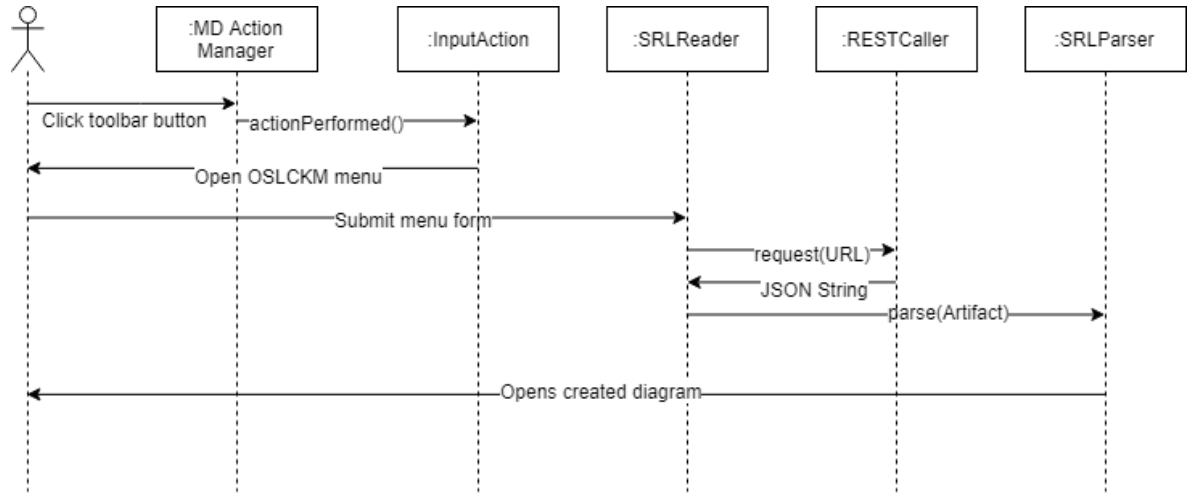

Figure 8: Load diagram sequence diagram

Figure 9 represents the request of all diagrams from an external OSCLKM interface to MagicDraw , which has a linear structure of functions, in which one class calls the next, and then all the information is returned in cascade.

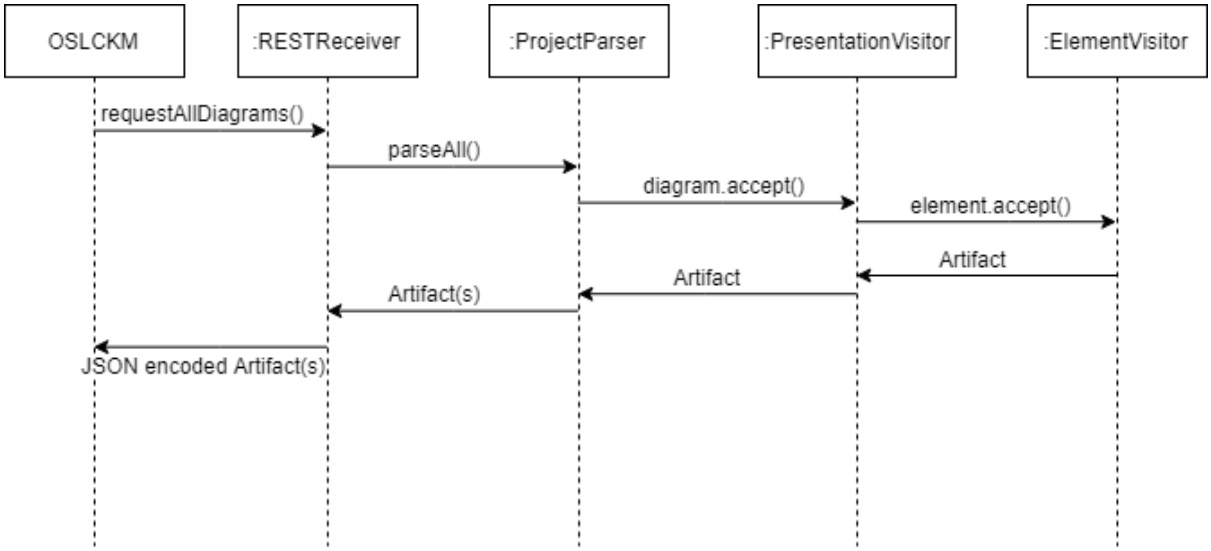

Figure 9: Request all diagrams sequence diagram

In the sequence diagrams represented in figures 10-13 the structure is similar to the one in figure 9, since we are still retrieving diagrams from MagicDraw, the only thing that changes is the filter applied in the ProjectParser class.

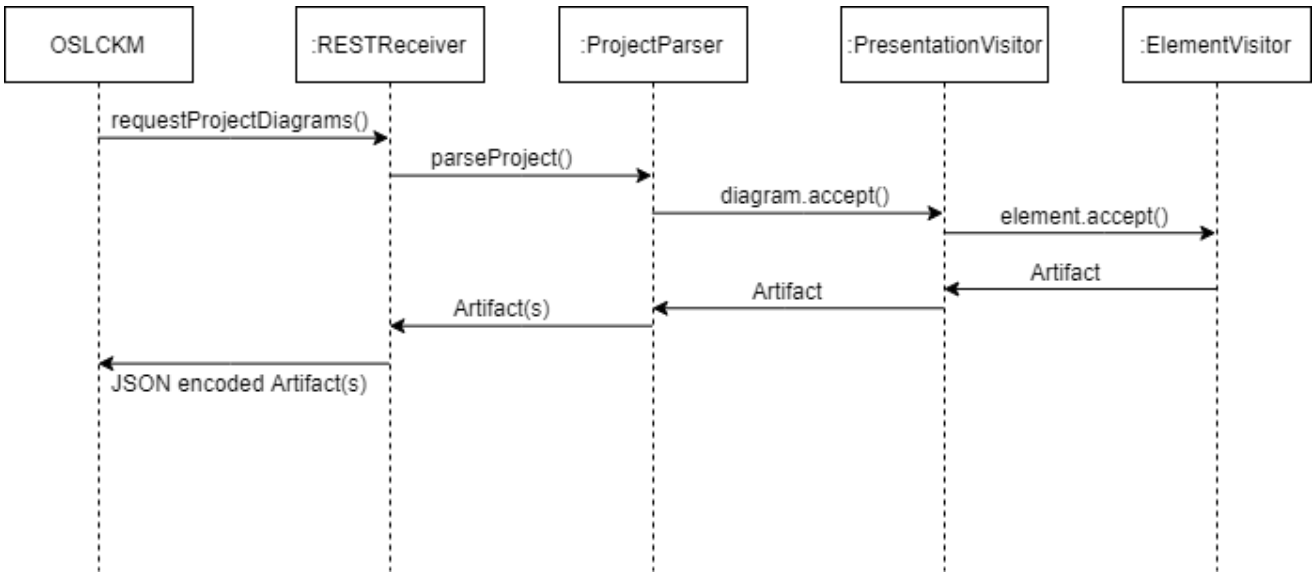

Figure 10: Request project diagrams sequence diagram

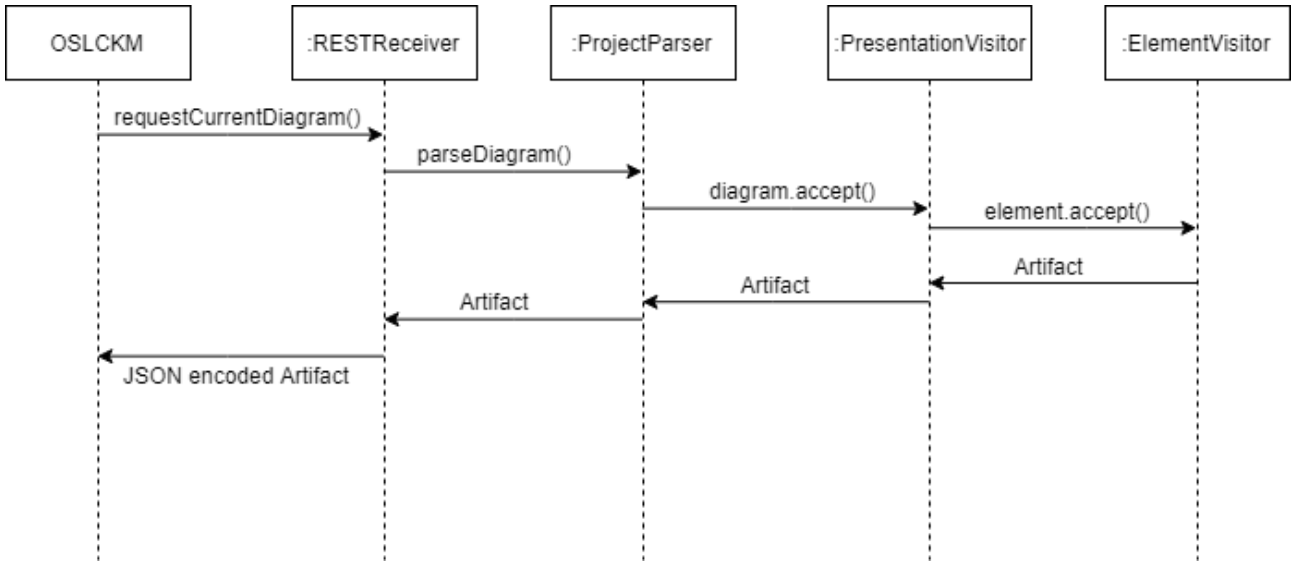

Figure 11: Request current diagram sequence diagram

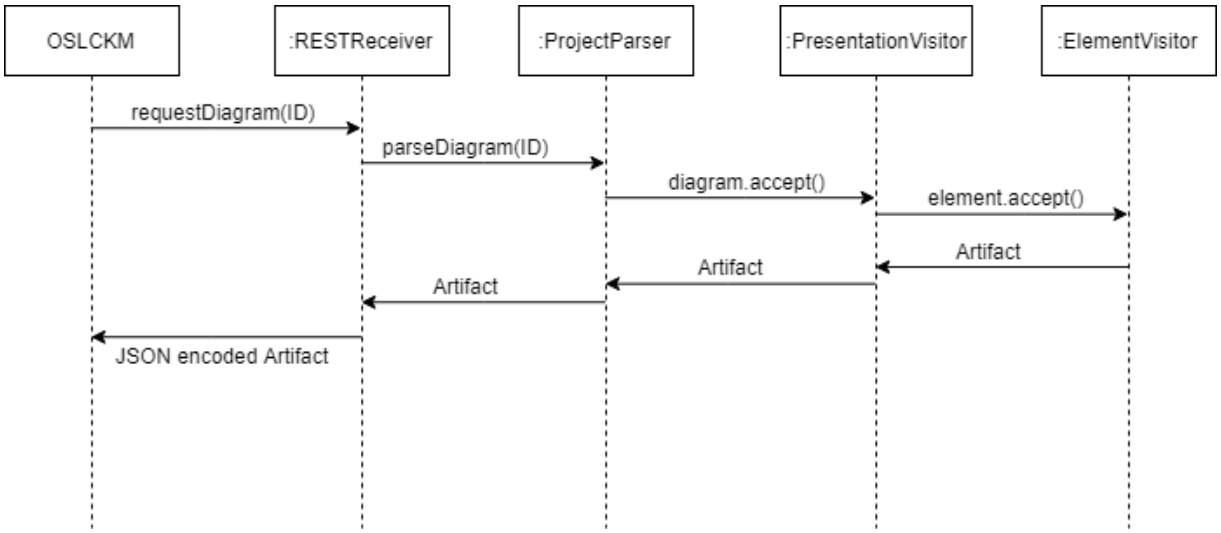

Figure 12: Request diagram by ID sequence diagram

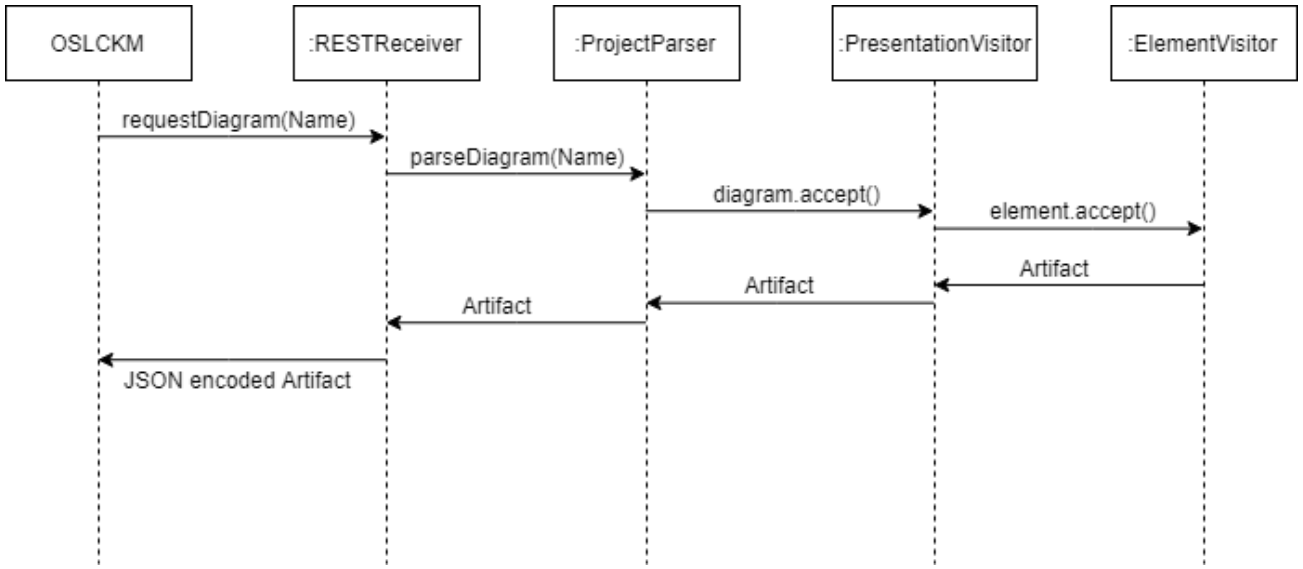

Figure 13:Request diagram by name sequence diagram

Figure 14 represents the most simple functionality path, showing a smaller sequence diagram. This is due to the fact that the functionality that must e implemented here has no need for web services, but will only store the JSON in the hard drive.

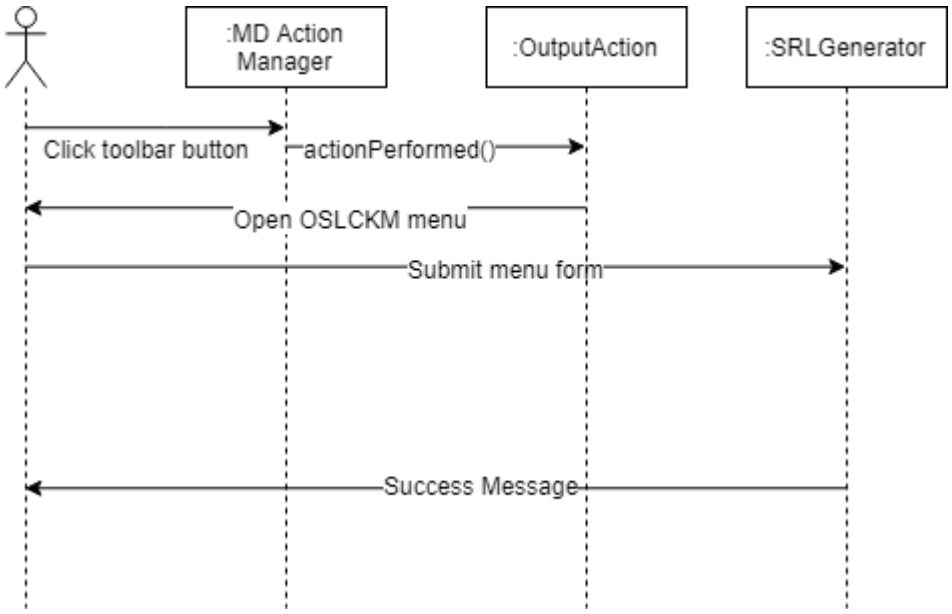

Figure 14: Export diagram to file sequence diagram

4.7 COMPONENT DIAGRAM

In this section the component diagram will be discussed. The different components displayed below are:

- C-1 MD Elements Manager: Component in charge of converting the elements created into real diagrams in MagicDraw and vice versa. It has a high usage of MagicDraw openAPI.
- C-2 MD Action Manager: Component that creates the interface buttons and menus, and is in charge of calling the necessary actions to be taken once the information is introduced by the user.
- C-3 SRL Generator: This component will convert the diagram elements obtained into OSLC objects.
- C-4 SRL Reader: This component will transform the OSLC objects received into diagram elements.
- C-5 OSLCKM: This component is in charge of the organization and serialization of the OSLC objects.
- C-6 REST Client: This component will take care of all the communications to be realized with OSLCKM interfaces.

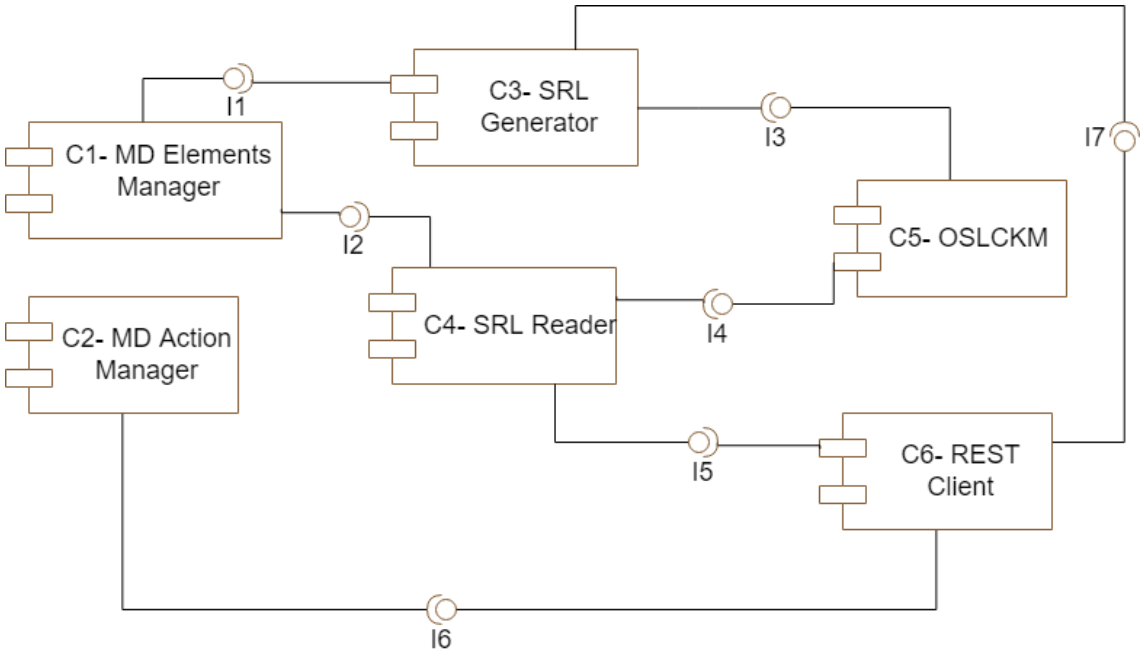

Figure 15: Component diagram

4.8 COMPONENT DESCRIPTION

Below, the description of each of the components that compose the system are displayed in a tabular fashion. Each of the tables represent one component, showing the following information:

- Identifier: unique identifier of the component.
- Type: Type of the component.
- Function: What role the component plays in the system.
- Subordinates: Which components depend of this component.
- Dependencies: Components this component is dependent of.
- Interfaces: interfaces this component provides.
- Resources: Resources the component needs.
- Processing: The operation the component carries out.
- Input: Input parameters of this component.
- Output: Output created by the component, for each case.

Identifier	C-1 MD Elements Manager	Type	Manager
Function	The function of this component will be reading and writing elements from the MagicDraw diagrams.		
Subordinates	C3 - SRL Generator, C4 - SRL Reader		
Dependencies	None		
Interfaces	I1, I2		
Resources	MagicDraw projects and diagrams		
Processing	Fetch and create diagrams		
Data	Input	Requested diagram(s)/Diagram contents	
	Output	Diagram contents/Created diagram	

Table 29: Description of Component 1

Identifier	C-2 MD Action Manager	Type	Manager
Function	The function of this component will be showing buttons in the MagicDraw toolbar and returning users' input. It is the graphical interface part of the plugin.		
Subordinates	None		
Dependencies	C6 - REST CALLER		
Interfaces	None		
Resources	Interface to be displayed		
Processing	Show and act on actions needed.		
Data	Input	GUI to be shown	
	Output	User input	

Table 30: Description of Component 2

Identifier	C-3 SRL Generator	Type	Subsystem
Function	The function of this component shall be the generation of SRL from diagram's content.		
Subordinates	None		
Dependencies	C1 - MD Elements Manager, C5 - OSLCKM, C6- REST Caller		
Interfaces	None		
Resources	Diagram contents, Transformation rules		
Processing	Parse diagram and convert it to SRL.		
Data	Input	Diagram contents	
	Output	One artifact per diagram parsed	

Table 31: Description of Component 3

Identifier	C-4 SRL Reader	Type	Subsystem
Function	The function of this component shall be the generation of diagram content from SRL		
Subordinates	None		
Dependencies	C1 - MD Elements Manager, C5 - OSLCKM		
Interfaces	I5		
Resources	SRL information, Transformation rules		
Processing	Parse SRL and transform it to diagram elements.		
Data	Input	SRL information	
	Output	Diagram elements	

Table 32: Description of Component 4

Identifier	C-5 OSLCKM	Type	Subsystem
Function	The function of this component shall be the processing of information to be in SRL format, as well as allowing SRL information to be interpreted.		
Subordinates	C3- SRL Generator, C4- SRL Reader		
Dependencies	None		
Interfaces	I3, I4		
Resources	OSLCKM classes		
Processing	Interpret SRL Resources, and create new ones.		
Data	Input	OSLCKM field inputs/SRL Artifacts	
	Output	SRL Artifacts/OSLCKM field inputs	

Table 33: Description of Component 5

Identifier	C-6 REST Client	Type	Subsystem
Function	This component will send and receive REST requests.		
Subordinates	C2 - MD Action Manager, C3- SRL Generator		
Dependencies	C4- SRL Reader		
Interfaces	I6, I7		
Resources	Messages to be sent and URLs to listen/request		
Processing	Interpret SRL Resources, and create new ones.		
Data	Input	URLs and messages to be sent	
	Output	SRL in JSON	

Table 34: Description of Component 6

4.9 COMPONENT TRACEABILITY

In this section a table will be presented showing which components are involved in fulfilling each of the system requirements established in previous sections. The requirement ID and name are provided, matched to each of the component ids.

System Requirement ID	Name	Component
SR-FR_01	Receive request for current diagram	C-6 REST Client
SR-FR_02	Receive request for current project	C-6 REST Client
SR-FR_03	Receive request for all diagrams	C-6 REST Client
SR-FR_04	Receive request for specific diagram(ID)	C-6 REST Client
SR-FR_05	Receive request for specific diagram(Name)	C-6 REST Client
SR-FR_06	Parse diagrams	C3 - SRL Generator
SR-FR_07	Filter diagrams by ID	C1 - MD Elements manager
SR-FR_08	Filter diagrams by Name	C1 - MD Elements manager
SR-FR_09	Display a button in the MagicDraw toolbar	C2 - MD Action Manager
SR-FR_10	Show a load menu	C2 - MD Action Manager
SR-FR_11	Check URL format validity	C2 - MD Action Manager
SR-FR_12	Request SRL diagram	C-6 REST Client
SR-FR_13	Parse SRL diagram	C4 - SRL Reader
SR-FR_14	Check diagram validity	C1 - MD Elements manager
SR-FR_15	Show an export menu	C2 - MD Action Manager

Table 35: Component Traceability Matrix

4.10 DEPLOYMENT DIAGRAM

The deployment diagram will represent the physical view of the application, that is, the different hardware components that are needed to implement the proposed solution. In the case of this plugin, all the functionality is in the same machine, and will all be included inside the MagicDraw environment. Nevertheless, there will be interaction with other systems in different clocks, but they will be outside of the implementation being presented here.

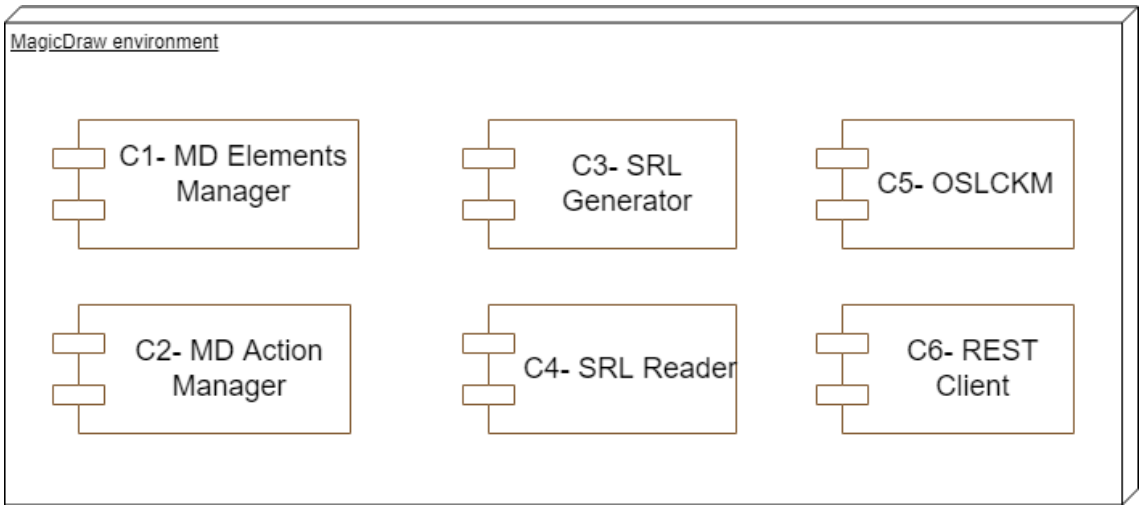

Figure 16: Deployment diagram

4.11 DATA TRANSFORMATION

In this section the process of transforming MagicDraw diagrams into SL (and vice versa) will be explained. The Magicdraw elements will be shown with their SRL counterpart in a table, explaining the hierarchy of elements in both cases. The table columns represent the following:

- **MagicDraw element**: Element in MagicDraw to be converted to SRL. Some elements will be considered as generic. (Ej: All inter-class relationships will be considered as one, since they have they carry the same information in the same fields). Please note that some of the fields included in a generic element are not applied to all of them.
- **SRL element**: Element name in SRL.
- **Parent SRL element**: Parent of the SRL element. In this case only the name and not the type.
- **SRL fields**: Fields that will be used in SRL to enclose information about the element.
- **SRL field content**: Content extracted from the element included in the SRL field.

Subsequently the generic MagicDraw elements used will be defined for further understanding of the transformation table:

- **Packageable element**: It is an element that can be owned directly by a package. This includes but is not restricted to classes, interfaces and other packages.
- **Relationship**: Relationships between classes, such as generalization, association, between others.

MagicDraw	SRL	SRL parent	SRL fields	MagicDraw content
Diagram	Artifact	None (Will be root element)	title	Diagram name
			type	“MD Diagram”
Packageable	Artifact	Diagram artifact or another packageable artifact	title	Element name
Packageable visibility	Metaproperty	Packageable artifact	tag	“Visibility”
			value	Element visibility
Packageable isAbstract	Metaproperty	Packageable artifact	tag	“Abstract”
			value	true/false

Packageable isActive	Metaproperty	Packageable artifact	tag	“Active”
			value	true/false
Packageable isLeaf	Metaproperty	Packageable artifact	tag	“Leaf”
			value	true/false
Packageable isFinalSpecialization	Metaproperty	Packageable artifact	tag	“Final Specification”
			value	true/false
Packageable property	Artifact	Packageable artifact	title	Property name
Property type	Metaproperty	Property artifact	tag	“Type”
			value	Property type
Property visibility	Metaproperty	Property artifact	tag	“Visibility”
			value	Property visibility
Property default value	Metaproperty	Property artifact	tag	“Default Value”
			value	Property default value (String)
Packageable operation	Artifact	Packageable artifact	title	Operation name
Operation type	Metaproperty	Operation artifact	tag	“Type”
			value	Method type
Operation visibility	Metaproperty	Operation artifact	tag	“Visibility”
			value	Method visibility
Packageable operation parameter	Artifact	Packageable operation artifact	title	Parameter name
Parameter type	Metaproperty	Parameter artifact	tag	“Type”
			value	Parameter type
Parameter visibility	Metaproperty	Parameter artifact	tag	“Visibility”
			value	Parameter visibility

Parameter default value	Metaproperty	Parameter artifact	tag	“Default Value”
			value	Parameter default value (String)
Packageable constraint	Artifact	Packageable artifact	title	Constraint name
			type	“Constraint”
Constraint specification	Metaproperty	Classifier artifact	tag	“Specification”
			value	Specification value (String)
Packageable port	Artifact	Packageable artifact	title	Port name
			type	“Port”
Packageable interface realization	Metaproperty	Packageable artifact	tag	Realized interface
			value	Will be another packageable artifact, with the same properties as the rest (Considering it is the definition of an interface).This artifact will therefore be duplicated, one where the interface belongs, and another here beneath the class with realizes the interface.
Relationship	RSHP	Diagram artifact	name	Relationship type
			from	From artifact
			to	To artifact
			type	Relationship type
Note	Artifact	Diagram artifact or packageable artifact	type	“note”
Note body	Metaprop	Note artifact	tag	“Body”

	erty		value	Note content
Anchor	RSHP	Diagram artifact	name	“Anchor”
Comment	Artifact	Diagram artifact or packageable artifact	type	“Comment”
Comment body	Metaprop erty	Comment artifact	tag	“Body”
			value	Comment content

Table 36: SRL-MagicDraw Elements Transformation

4.12 USER INTERFACE

Considering the requirements defined previously, there is no need for much interface, other than a menu to select a URL, and one to export the diagram into a file. There is also two buttons defined as necessary, but it is established that they will contain only text, and therefore are not really part of a new interface, but part of MagicDraw's native interface.

4.12.1 LOAD MENU

The menu described in SR-FR_10 will contain two interfaces. The first one will ask for the service provider URL and the security settings to connect to it, and the second will request the type of artifact to be created and the operation to perform on it.

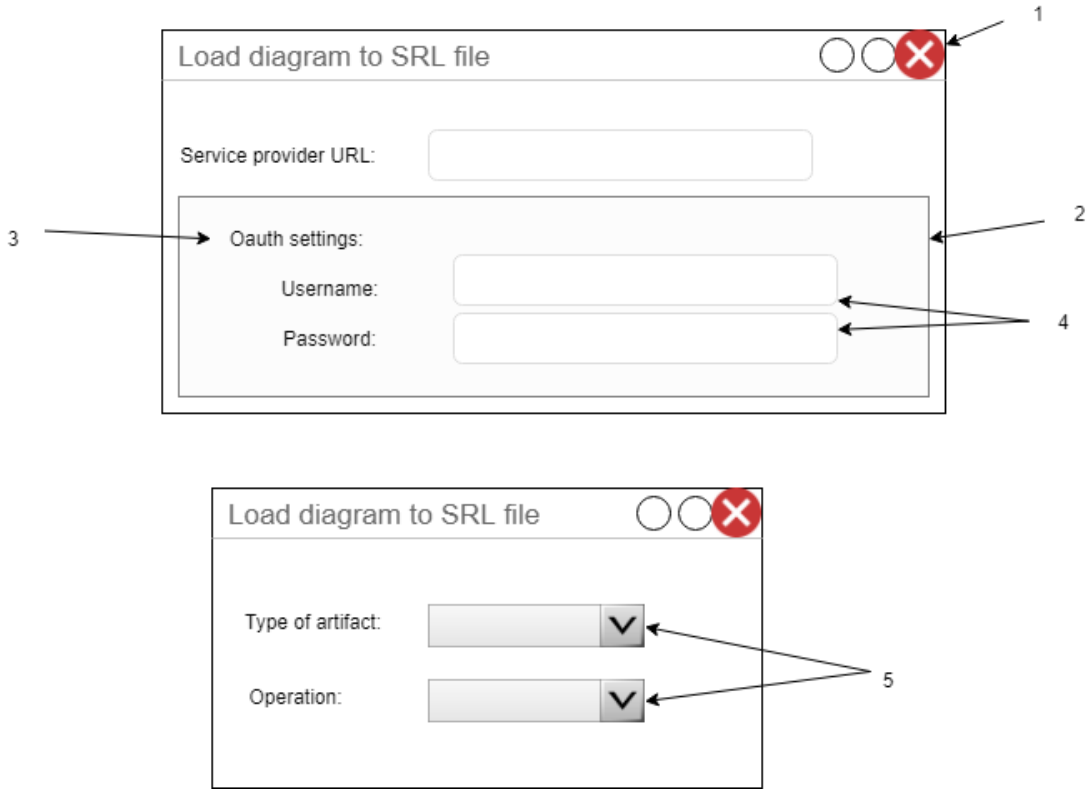

Figure 17: Load menu mockup

1. Close button to cancel the operation.
2. Section to divide the security settings: This will provide the user context so he has better understanding of what data he is filling in.
3. Section label: It will tell the user what context is the section used for.
4. Input fields: Fields for the user to fill in the data. The password field will show a dot character instead of its contents as it is usually done with password fields.

4.12.2 EXPORT MENU

The menu described in SR-FR_15 will be fairly simple, in a form fashion inside a modal window. It will have to accommodate a text input field for the user to include the file path location (As well as display it if the user chooses to use the file picker interface to choose a file) and a well visible button to confirm the path has been already filled in, as well as a button to go back.

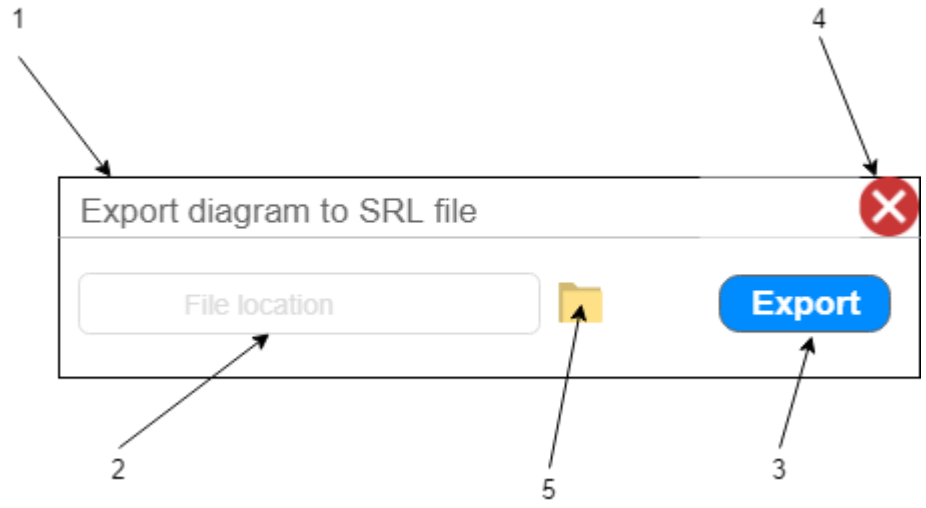

Figure 18: Export menu mockup

1. Window title bar: Includes the action being performed to make sure the user does not get lost in the action or makes a mistake.
2. Text input field: Has the text “File location” as placeholder to help the user understand the data that has to be filled in.
3. Export button: In a distinguishable color as blue, that can be related to going forward with the action. It is placed in the right side of the window since the target audience reads from left to right and it will be the last thing that needs to be used.
4. Close button: It will allow the user to exit if he decides he does not want to go through with the action. It is displayed in red, a color associated with canceling an action.
5. Folder Icon: it will allow the user to browse through different files in the computer.

5. IMPLEMENTATION & TESTING

Now, the methods used for the implementation of the software will be exposed, together with the difficulties found in each of the phases that were involved in the development of the implementation.

5.1 SUPPORT TOOLS USED

In order to produce the code that will fulfill the necessary requirements, some support programs to help with development will be used.

First of all, eclipse IDE will be used to implement the Java code. MagicDraw provides an initial project to work with by default in "<installation_directory>/openapi/IDE/eclipse". In this project all the libraries are already imported and it is possible to use eclipse debug mode to open MagicDraw and update the plugin code on the go. Nevertheless, after thorough research a way to make this work was not found. Therefore, as a substitute for debugging, the project was exported as a jar directly into the MagicDraw installation directory and magicdraw reopened to load the new version of the plugin.

Another program that was used was Notepad++. Notepad++ was used to create the file plugin.xml, needed as part of the installation of the plugin. Notepad++ was also used for the purpose of debugging SRL in JSON format, together with a plugin, JSTool. This program allowed to comfortably format the JSON contents created by the plugin with a simple command, making them readable. Not only that, but it also allowed the JSON to be seen in a tree fashion, as shown below.

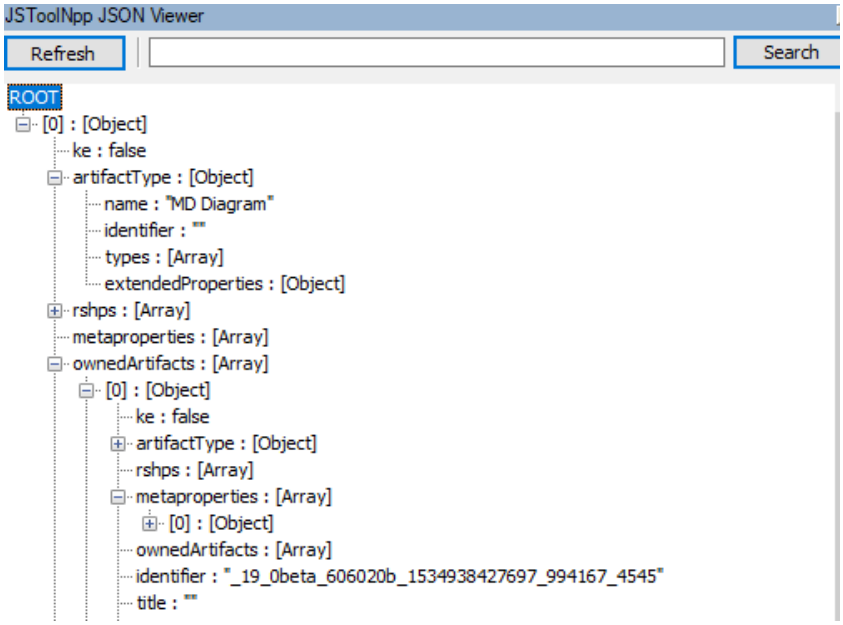

Figure 19: Notepad++ JSTool tree view

5.2 METHODOLOGY

For the implementation of the OSLCKM plugin, agile methods were used, in the sense that it was gradually implemented in smaller functional applications. This is, smaller, complete functionalities were implemented one by one, making sure they worked properly before advancing to the next functionality. Each one of these phases taken will be named sprints in this document, even though they do not follow the traditional fixed-time format they traditionally follow. In this manner, the functional units that were developed were the following(Each sprint includes all the previous functionality, plus the described in it):

1. Create a working plugin with access to the data model.
2. Generate SRL including only blocks and relationships.
3. Generate SRL including the complete diagram.
4. Obtain a complete diagram from SRL.
5. Create a menu to choose URL.
6. Implement the web service and calls to other web services

The agile framework that the methodology was based on is SCRUM. SCRUM defines a working scheme in which four different parts are involved[11]:

- Stakeholders: not present in the adaptation used, since it is not a solution that has been funded, although we might consider that the end users of the plugin can be included in this group(The problem is that they were not involved in decision taking).
- Product owner: It is the person responsible of guiding the project, delimiting the requirements(Or as it is called in scrum, product backlog). In the adaptation used this role would correspond to Jose María, since he was the teacher that validated my work.
- Development team: This is the group of multi-disciplinary people (or person in this case) that develop the solution. In this case it would be me.

During scrum, there is an initial planning in which the work and sprints are defined, and then each sprint has the following phases:

- Sprint planning: In this phase the amount of work that must be done during the sprint(sprint backlog) is defined, and divided in tasks.
- Development: Here the sprint backlog is developed. During this phase there is a short daily meeting to sync the development team called “Daily SCRUM”.
- Sprint review: The work done during the sprint is shown to the product owner(and optionally stakeholders) for validation.
- Retrospective: This is an analysis of what has been done correctly and should be kept and what has to change for future sprints. It is the core of agile methods, since it creates a situation of constant improvement if done correctly.

5.3 SPRINTS

Following, each sprint will be broken down, explaining in more detail the functionalities developed and the work carried out.

5.3.1 SPRINT ONE

In this sprint the focus was creating a functional plugin, with no real useful functionality, but with all libraries imported to make sure the initial design of the system was feasible. It includes the creation of a project with a class extending MagicDraw's Plugin class, that has imported all MagicDraw's libraries, as well as the OSLCKM library. As well during this sprint, the menu buttons that are required in according to SR-FR_10 and SR-FR_15 were implemented, with no functionality when clicking them.

During this sprint a problem arose, the OSLCKM library was generating runtime exceptions when declaring variables of types included in the library. After some investigation and external help, it was discovered that there were some extra dependencies for the OSLCKM library. The extra jars were included in the plugin folder and the plugin.xml file was updated accordingly.

5.3.2 SPRINT TWO

Sprint two was aimed to obtain the first piece of complete functionality. The diagrams opened in MagicDraw will be parsed to SRL, but will be stored in a file for debugging for now instead of providing a web service, since it was enough to check the parsing of the diagram.

5.3.3 SPRINT THREE

In sprint three the functionality implemented in sprint two is expanded by including every element in the diagram this time. Some elements that were not included previously were notes, since they are included in the MagicDraw openAPI as presentation elements only, and not regular elements. They also compromised the information included inside other elements such as clases. Some of these elements had elements themselves like the parameters included inside a class' operations.

The main challenge in this area was collecting every piece of information obtained and providing a suitable place to store it in the OSLC data model. In order to accomplish this OSLC metaproperties were used to store extra pieces of information attached to the artifacts.

5.3.4 SPRINT FOUR

In sprint four the functionality to obtain MagicDraw diagrams is started. Since in sprints two and three all the different pieces of information contained in the diagram had been examined, this sprint will cover the creation of a complete diagram in one go, without dividing it like before.

During this sprint there was also a blocking problem that caused delays. When adding an element to the created diagram, a runtime exception would be thrown, claiming no container

was found for the element in the diagram. After some help from MagicDraw's support, the problem was solved by creating a shape for each of the diagram's elements and adding the shape instead of the element itself.

5.3.5 SPRINT FIVE

Sprint five was short in development time. Using the native Java JFrame the pop-up menus were created to show the desired menus, as the ones designed shown previously in this document. The interface was anchored to the buttons created in sprint 1 and the functionalities from other sprints, that had been created with static parameters.

The URL format checker in the load interface was created and tested, but disabled, since the web services were not yet available at this point. As a substitute, a file path with the JSON content was provided in the input textbox to test the implemented functionalities. The URL box will contain a file path for testing purposes, parsing the contents of the file instead of a web returned JSON.

5.3.6 SPRINT SIX

Finally during the final sprint, the web services used for receiving diagram requests were implemented, together with the implementation of calls to URLs requested by the user through the menu. The implementation of all the functionality was already done, so it only needed to substitute the read/write from files to exchange of JSON through web services.

5.4 TESTING

5.4.1 TEST CASES

Since the plugin created needs the MagicDraw application environment in order to work, automated tests were not possible to be created. Therefore, manual tests were established as a way to ensure the quality of the created plugin. The tests will be exhibited in the following table, representing their assigned IDs, description, functionality being tested and the expected result.

ID	TC_01
Component	C-2 MD Action Manager
Description	Make sure the component is loading in MagicDraw.
Procedure	The plugin should be installed in MagicDraw under the directory <MagicDraw installation dir>/plugins/OSLCKMPlugin containing in that folder the following files: ● OSLCKMPlugin.jar ● Plugin.xml ● Oslc4j-core-2.0.0.jar ● oslc4j-core-2.1.0.jar
Prerequisites	MagicDraw must be installed in the machine.
Input	All required files
Output	Success: The plugin loads properly. This can be checked by looking at the "Import SRL Diagram" button that should appear. Furthermore, the MagicDraw logs must not show any errors regarding the plugin.
	Fail: The plugin does not load (The button does not appear) or loads with errors.

Table 37: Test Case 1

ID	TC_02
Component	C-6 REST Client
Description	Verify the web service at the desired URL is available. This will be done through an external program such as Postman. The URLs that must be checked are the following: ● http://localhost:/
Procedure	For each of the URLs an HTTP request must be sent in order to check the response code obtained. At this point the content of the request and the response are not checked.
Prerequisites	Have the plugin installed and MagicDraw running in the same machine.
Input	URLs
Output	Success: The HTTP request sent is answered with status code 200 (OK).
	Fail: The request either does not receive a response o receives one with a status code different that 200.

Table 38: Test Case 2

ID	TC_03
Component	C-1 MD Elements Manager, C-3 SRL Generator, C-5 OSLCKM
Description	Corroborate that the plugin returns a JSON with all open diagrams when requested.
Procedure	Using an external program such as Postman make a GET request.
Prerequisites	Have the plugin installed and MagicDraw running in the same machine with several projects open.
Input	URL
Output	Success: The HTTP request sent is answered with status code 200 and the correct JSON is received.
	Fail: The request either does not receive a response o receives one with a status code different that 200, or an incorrect JSON is received.

Table 39: Test Case 3

ID	TC_04
Component	C-1 MD Elements Manager, C-3 SRL Generator, C-5 OSLCKM
Description	Corroborate that the plugin returns a JSON with all diagrams in the currently open project.
Procedure	Using an external program such as Postman make a GET request at .
Prerequisites	Have the plugin installed and MagicDraw running in the same machine with at least one project with more than one diagram open.
Input	URL
Output	Success: The HTTP request sent is answered with status code 200 and the correct JSON is received.
	Fail: The request either does not receive a response o receives one with a status code different that 200, or an incorrect JSON is received.

Table 40: Test Case 4

ID	TC_05
Component	C-1 MD Elements Manager, C-3 SRL Generator, C-5 OSLCKM
Description	Corroborate that the plugin returns a JSON with currently open diagram.
Procedure	Using an external program such as Postman make a GET request.
Prerequisites	Have the plugin installed and MagicDraw running in the same machine with several projects open (each with a few diagrams open).
Input	URL
Output	Success: The HTTP request sent is answered with status code 200 and the correct JSON is received.
	Fail: The request either does not receive a response o receives one with a status code different that 200, or an incorrect JSON is received.

Table 41: Test Case 5

ID	TC_06
Component	C-1 MD Elements Manager, C-3 SRL Generator, C-5 OSLCKM
Description	Check the plugin is properly filtering the diagrams by ID.
Procedure	Using an external program such as Postman make a GET request.
Prerequisites	Have the plugin installed and MagicDraw running in the same machine with several projects open (each with a few diagrams open).
Input	URL
Output	Success: The HTTP request sent is answered with status code 200 and the correct JSON is received.
	Fail: The request either does not receive a response o receives one with a status code different that 200, or an incorrect JSON is received.

Table 42: Test Case 6

ID	TC_07
Component	
Description	Check the plugin is properly filtering the diagrams by name.
Procedure	Using an external program such as Postman make a GET request.
Prerequisites	Have the plugin installed and MagicDraw running in the same machine with several projects open (each with a few diagrams open).
Input	URL
Output	Success: The HTTP request sent is answered with status code 200 and the correct JSON is received.
	Fail: The request either does not receive a response o receives one with a status code different that 200, or an incorrect JSON is received.

Table 43: Test Case 7

ID	TC_08
Component	C-2 MD Action Manager, C-4 SRL Reader, C-5 OSLCKM, C-6 REST Client
Description	Check the menu is working(at a technical level).
Procedure	Open the menu, fill in the data and check that the diagram is created accordingly. Try with different diagrams with different elements in them.
Prerequisites	Have the plugin installed and MagicDraw running in the same machine with a project open.
Input	JSON at URL
Output	Success: The diagram is created successfully in the current project and matches the target JSON requested.
	Fail: Either the request fails, the diagram is not able to be created, or the creation of the diagram is incorrect.

Table 44: Test Case 8

ID	TC_09
Component	C-2 MD Action Manager
Description	Check the menus have an appropriate user-friendly interface.
Procedure	This can be tested by making sure everything is properly labeled and sized and also by obtaining external feedback.
Prerequisites	Have the plugin installed and MagicDraw running in the same machine with a project open.(The button to open the menu will be disabled otherwise.)
Input	URL
Output	Success: User interface is user-friendly.
	Fail: User interface is not appropriate.

Table 45: Test Case 9

ID	TC_10
Component	All components
Description	Check the requests have an appropriate duration.
Procedure	Using an automated script, measure the time taken for each request at least 5 times. Obtain the average of each request and make sure the time taken is below the duration specified in SR-NFR_01.
Prerequisites	Have the plugin installed and MagicDraw running in the same machine. Have all prior tests passed to make sure the measured time is of the correct functionality.
Input	URL
Output	Success: The time is below the specified.
	Fail: The time is above the specified.

Table 46: Test Case 10

ID	TC_11
Component	C1- MD Elements Manager, C2- MD Action Manager, C3- SRL Generator, C5- OSLCKM
Description	Check the OSLCKM export is working properly
Procedure	Click on the OSLCKM export button, and then choose a file to export to. Check the result.
Prerequisites	Have several diagrams open.
Input	Diagram, destination file
Output	Success: The file is created properly.
	Fail: The file is not created or the contents is incorrect.

Table 47: Test Case 11

5.4.2 TEST TRACEABILITY MATRIX

The test traceability matrix is used to make sure every defined functionality of the system has been tested, making sure the quality of the product is kept.

	TC_01	TC_02	TC_03	TC_04	TC_05	TC_06	TC_07	TC_08	TC_09	TC_10	TC_11
SR-FR_1					X						
SR-FR_2		X		X							
SR-FR_3		X	X								
SR-FR_4		X				X					
SR-FR_5		X					X				
SR-FR_6		X									
SR-FR_7						X					
SR-FR_8							X				
SR-FR_9	X							X	X		
SR-FR_10								X	X		
SR-FR_11											
SR-FR_12											
SR-FR_13											
SR-FR_14											
SR-FR_15											X
SR-FR_16											X
SR-NFR_01										X	
SR-NFR_02	X										
SR-NFR_03									X		

Table 48: Test Traceability Matrix

6. BUDGET AND PLANNING

In this chapter the project's budget and planification will be broken down. In order to properly explain this process, a table with the amount of hours dedicated to each phase of the project, a Gantt diagram and budget breakdown are shown hereafter.

6.1 TIME PLANNING

The project duration will be divided in phases by hours. Each phase will be divided itself by the different tasks that compose them.

Phase	**Tasks**	**Description**	**#Hours**
Project definition	Define project	Define the software to be built.	3
	Define scope	Detail all the functionality, with its reach and limits	7
	Total phase hours		**10**
Research	State of the art	Research about the current system in use	16
	MagicDraw	Find MagicDraw's openapi's features and limitations, as well as the way to obtain the maximum amount of diagram information possible.	8
	OSLCKM	Investigate the API, finding the different fields to store information.	13
	Java Interfaces	Figure out how Java's JFrame system works.	5
	Total phase hours		**42**
Analysis	User stories	Define user stories and the associated use cases.	3
	User requirements	Describe user requirements	8
	System functional requirements	Determine the functional requirements.	15

	System non-functional requirements	Define the application's non-functional requirements.	10
	Total phase hours		**36**
Design	Class diagram	Definition of the class diagram.	8
	Sequence diagrams	Creation of sequence diagrams.	10
	Component diagram	Construct the component diagram.	5
	Deployment Diagram	Form the deployment diagram.	5
	Transformation design	Design on the transformation of diagrams to SRL(and vice versa).	10
	Interface design	Design the interface of the menu pop-up of SR-FR_10, considering both SR-FR_10 and SR-NFR_03.	5
	Total phase hours		**43**
Implementation	Sprint 1	Content defined in 5.2Methodology	10
	Sprint 2		20
	Sprint 3		22
	Sprint 4		27
	Sprint 5		12
	Sprint 6		6
	Total phase hours		**97**
Testing	Testing	Execution and refinement of the defined tests.	**22**
Documentation	Definition	Decide the contents to be included in the document.	7

	Redaction	Writing the initial draft of the document.	37
	Corrections	Corrections of mistakes and adding missing items.	10
	Total phase hours		**54**
TOTAL (in hours)	Total project hours		**304**

Table 49: Project Hours Computation

6.2 GANTT DIAGRAM

Below the Gantt diagram is presented. As it can be observed we can view two different mayor processes.

First, the development of the code can bee seen in the implementation part. It can be observed that as it will be mentioned, the sprints used are of uneven length. Most of them are done back to back, but the first one is done prior to the rest and separated, since it was really part of the research done in magicdraw and OSLCKM APIs.

Second of all, we can see all the overhead of the product produced. From the definition of the project to the definition of the requirements we can see how they have been done back to back, with some overlaps, providing a staircase shape.

		May				June				July				August				September			
		W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	
Project definition	Define project																				
	Define scope																				
Research	State of the art																				
	MagicDraw																				
	OSLCKM																				
	Java Interfaces																				
Analysis	User stories																				
	User requirements																				
	Functional requirements																				
	Non-functional requirements																				
Design	Class diagram																				
	Sequence diagrams																				
	Component diagram																				
	Deployment Diagram																				
	Transformation design																				
	Interface design																				
Implementation	Sprint 1																				
	Sprint 2																				
	Sprint 3																				
	Sprint 4																				
	Sprint 5																				
	Sprint 6																				
Testing	Testing																				
Documentation	Definition																				
	Redaction																				
	Corrections																				

Table 50: Gantt Diagram

6.3 BUDGET BREAKDOWN

In this section the budget that was required for the completion of the project will be reviewed. In order to provide a better understanding of the expenses the production of the application caused, the budget will be presented in different tables, with a final summary at the end.

6.3.1 Direct Costs

The direct costs will be the sum of the costs of human resources, equipment, licensing and consumables.

To calculate the human resources costs, the salary established for the development will be 21€/hour.

Human Resources		
Phase	Hours	Costs (€)
Project definition	10	126
Research	42	630
Analysis	36	357
Design	43	546
Implementation	97	1155
Testing	22	210
Documentation	54	441
TOTAL	304 h	6384 €

Table 51: Human Resources Costs

For the computation of the costs of equipment, the wear of the computer will be taken into account, for the duration of the project. For this, we will take as reference an average of 3 years of a computer's lifespan(In good enough condition to make sure development is not delayed) will be used, and a price of 1050€ per computer. From these assumptions, we can obtain the following formula:

$$Computer\ wear\ = \frac{price}{lifespan(in\ months)} * number\ of\ development\ months =$$ equipment cost

Applying this formula we can obtain a total of 146.00€.

During the duration of the project, eclipse will be used for coding.Eclipse is free for all uses, so no costs will come from that license. Nevertheless, since the product is developed in MagicDraw, a license will be required for the duration of the project. The minimum license of MagicDraw is 1 year, and will cost $1600, equivalent to 1370€.

Regarding consumables, several office supplies will be needed for organization and work optimization:

Consumables	Paper	Post-it	Pens	Toner	**TOTAL**
Quantity	100 paper sheets	2 blocks	5 pens	1 package	
Price(€)	5	11	1	20	37€

Table 52: Consumables Costs

Direct Costs Summary	
Human Resources	6384€
Equipment	146 €
Licensing	1370 €
Consumables	37 €
TOTAL	**7937€**

Table 53: Direct Costs Summary

6.3.2 Indirect Costs

The indirect costs will be 15% of the already calculated direct costs, totaling to 1190.55€.

6.3.3 Budget Summary

With both direct and indirect costs, total costs can be calculated. Profit of 15% will be added, as well as taxes, included under general IVA tax (21%).

Total Budget Summary	
Direct costs	7937 €
Indirect costs	1190.54 €
Total costs	9127.54 €
Return on investment (10%)	912.75 €
Total before taxes (21%)	10040.29 €
Taxes	2108.46 €
TOTAL	**10231.97 €**

Table 54: Budget Summary

7. SOCIO-ECONOMIC ENVIRONMENT AND LEGAL FRAMEWORK

7.1 ECONOMIC IMPACT

Knowledge is a big part of the resources a company has, and therefore proper management of it will create a big impact. Without the ability to reuse this generated knowledge, many hours of work that were invested in a certain project have to be redone when keeping maintenance on such product or creating a new version of it.

Through this project, reuse of data and operations is created through interoperability, creating a better data structure for the new business operation model. This new model, named "Industry 4.0"[12]. This is the current fourth industrial revolution, which is characterized by the uprising of new technologies such as AI and IoT. In this new model of business there is access to huge amounts of data in real time from many different sources, each of which providing different data formats.

This project will also cover the concept "Collaborative and continuous engineering"[13], in which, in order to cut back in time effort, and therefore cost, engineering work is reused between different projects to allow speeding up development processes and saving unnecessary work that has already been done.

Overall, we can see two main impacts in the economic environment. The enablement of Industry 4.0, which will help business organizations adjust and learn using the data that is received from these new technologies, producing an increase in sales. Also, making use of the reuse capabilities of this new data management system, time schedules can be shortened, affecting directly on the development costs.

7.2 LEGAL IMPLICATIONS

The legal implications that affect the development of this software are the licenses used. This includes both the license of MagicDraw to develop the plugin, as well as the license of all the programs used in course of the development to aid in it and improve the development speed.

7.2.1 MAGICDRAW LICENSING

Regarding the licenses needed for magicdraw, we must distinguish two different types of MagicDraw usage: MagicDraw use in order to access the openAPI, and MagicDraw as a modeling tool, used for the purpose of testing the plugin's functionality.

On first place, the openAPI library MagicDraw provide to extend its functionality is free of use. This allows any developer to create and distribute the created plugins. This enables the creation of a community around MagicDraw that will create different types of plugins, improving its functionalities, and therefore encouraging people to use this tool.

Concerning the license of the use intended for the testing of the plugin, since the use of the software is not for commercial use itself, that is, it will not be used to create diagrams for the

obtainment of revenue, the demo version would normally be sufficient. Nevertheless, considering the small duration of the demo version (20 days), it would not last so a standard license of the program was needed.[14]

7.2.2 ECLIPSE LICENSE

Eclipse was used in the development for the production of Java code. The Eclipse software is provided under a public license for all uses. This license offers some restrictions, but only when extending functionality for commercial use. Therefore, there is no real implication in the use of this tool. The licensing document[15] does contain a disclaimer to exhumate Eclipse from all liabilities, but since once all the code is finished it will be completely detached from Eclipse, it is not a concern.

7.2.3 NOTEPAD++ LICENSE

Notepad++ is licensed under GLP, providing free use, including for commercial uses. Once again, for the creation of plugins, there are restrictions to control that the native notepad++ is not distributed as a paid tool[16]. This is not a concern since no additional functionality is being created for notepad++.

8. CONCLUSIONS AND FUTURE WORK

Throughout the course of this project the benefits of interoperability were analysed and found to improve in great measure the revenue by decreasing the amount of time needed to repeat certain tasks. Nevertheless, when implementing the solution that was proposed in this document, the cost of making interoperability realized has also to be considered. There is no real way of obtaining it without the implementation of custom solutions for each tool, making it necessary to implement for each tool. Even considering this fact, i can conclude confidently that interoperability will still reduce costs considerably in the long run, and therefore improve revenues in the current industry.

On a personal level, several new concepts and use of technologies were learnt during this project. First of all, the concept of knowledge management was new to me prior to the production of this project. Throughout the development I was able to learn what it really meant, as well as the current state it is in and where it is going to. The different techniques used nowadays were learnt, together with the existing problems. These problems were analyzed with the objective of obtaining a better understanding of what it entailed to implement interoperability in a system.

Not only that, but during the implementation of the proposed solution, I also learnt some new technology related skills, as well as improved some existing ones. The amount of work I had done prior to this project manipulating data in an iterative way was limited. It was also the first time I used the GSON library to transform objects into, and from JSON; with this library it was simple to serialize data to enable its transmission through the web.Also, this project was used as a way of seeing the whole process of a software product development, consolidating existing knowledge I had.

During the beginning of this project, some objectives were set that were desired to be completed as part of it. One if the objectives proposed was to explore the knowledge management area, which I think it was completed successfully, since the amount of familiarity with this content that I have aquired is quite wide, considering the duration of the project. The current state of interoperability was learnt, together with the problems that came with it, and how it can be solved (although it needs some investment). The objective of providing a tool to transfer data from MagicDraw in way that can be interpreted by other systems was clearly fulfilled, although it is true that the reuse of operations provided was limited.The quality assurance of the implementation could not be through automatic processes how I intended at first, but I was able to test in thoroughly through manual tests nonetheless.

With all the exposed points, I can conclude overall that the project was a success, since all objectives were fulfilled, even though some only to some extent, and the amount of knowledge I carry about the subject leaving this project is much greater than the one possessed before.

8.1 FUTURE WORK

Even though the objectives of this project were completed, there is much more work to do to connect more programs to this way of reusing data and operations, by implementing solutions for them. Not only that, but inside MagicDraw there is still room for improvement regarding operations.

In order to be able to produce interoperability, there needs to be a great variety of programs offering OSLCKM interfaces, in order to be able to use the produced information in different environments. Without this, it makes no sense to produce a standardized data output, since there is no client to use it. This is not the first program to have this interface, but many programs used in engineering are still in need for implementations to standardize their data.

Regarding the objectives set the work surrounding MagicDraw operations, the reason it was not implemented in this project was the lack of provision of allowing the access to operations from the openAPI. It was possible to implement the diagram validation operation through the creation of the diagram, since when the edit session is finished, MagicDraw´s openAPI automatically validates de diagram, exposing any inconsistencies. No other operations were found available for openAPI use, but in future versions of the API they will hopefully be added.

9. BIBLIOGRAPHY

[1] OSLC KM-Knowledge Management (n.d.). Retrieved June 10, 2018, from http://trc-research.github.io/spec/km/

[2] O. Signore, "Representing Knowledge in the Semantic Web", W3C, Italy, Paper, 2005, from http://www.w3c.it/papers/openCulture2005.pdf

[3] W3C (2004, February 10), Resource Description Framework (RDF): Concepts and Abstract Syntax. Retrieved September 20, from https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[4] W3C (2014, February 25), RDF Schema 1.1. Retrieved September 20, from

https://www.w3.org/TR/rdf-schema/

[5] W3C (2004, February 10), OWL Web Ontology Language Reference. Retrieved September 21, from https://www.w3.org/TR/2004/REC-owl-ref-20040210/

[6] W3C (2013, February 5), RIF Basic Logic Dialect (Second Edition). Retrieved September 21, from https://www.w3.org/TR/rif-bld/

[7] J. Llorens, J. Morato, and G. Genova, "RSHP: an information representation model based on relationships," in Soft Computing in Software Engineering, vol. 159, E. Damiani, M. Madravio, and L. Jain, Eds. Springer Berlin Heidelberg, 2004, 221–253.

[8] Overview of OSLC (n.d.). Retrieved September 5, 2018, from http://open-services.net/resources/tutorials/integrating-products-with-oslc/overview-of-oslc/

[9] J. Rumbaugh, I. Jacobson, G. Booch, *The Unified Modeling Language User Guide*, 2nd edition. Addison-Wesley, 2005, 496.

[10] Source Making, Visitor Design Pattern (n.d.). Retrieved August 5, 2018, from https://sourcemaking.com/design_patterns/visitor

[11] J.Sutherland and JJ Sutherland, Scrum: The Art of Doing Twice the Work in Half the Time, 1sted.New York: Crown Business, 2014.

[12] Deloitte, "¿Qué es la Industria 4.0?", Deloitte. [Online]. Available at: https://www2.deloitte.com/es/es/pages/manufacturing/articles/que-es-la-industria-4.0.html

[13] A. Goldberg, "Collaborative Software Engineering", *The Journal of Object Technology*, vol. 1, n.º 1, pp. 1-19, May 2002.

[14] MagicDraw UML v 6.0 Pricing and Licensing (n.d.). Retrieved September 13, 2018, from http://www.nomagic.com/files/60PricingPublicBrochure.pdf

[15] Eclipse Public License - v 1.0 (n.d.). Retrieved September 13, 2018, from https://www.eclipse.org/legal/epl-v10.html

[16] Notepad++ 6.1.1 - License (GPL) enhanced (n.d.). Retrieved September 13, 2018, from https://notepad-plus-plus.org/news/notepad-6.1.1-gpl-enhancement.html